*Submitted by*

*BHAVIN P V*

**Contents**

[**1. SVN**](#_dpkj42slsbvf) **Basics 3**

[1.1 Introduction](#_8vh1hvo89jhx) 3

[1.2 Basic](#_77by8wvhto8z) Uses 4

[**2. Building**](#_mulmf51vnbmz) **, Flashing and Debugging** 10

[2.1 Building](#_h62i1bumb12c) 10

[2.2 Flashing](#_juhzq9dv4ds4) and Debugging 12

**CHAPTER 1**

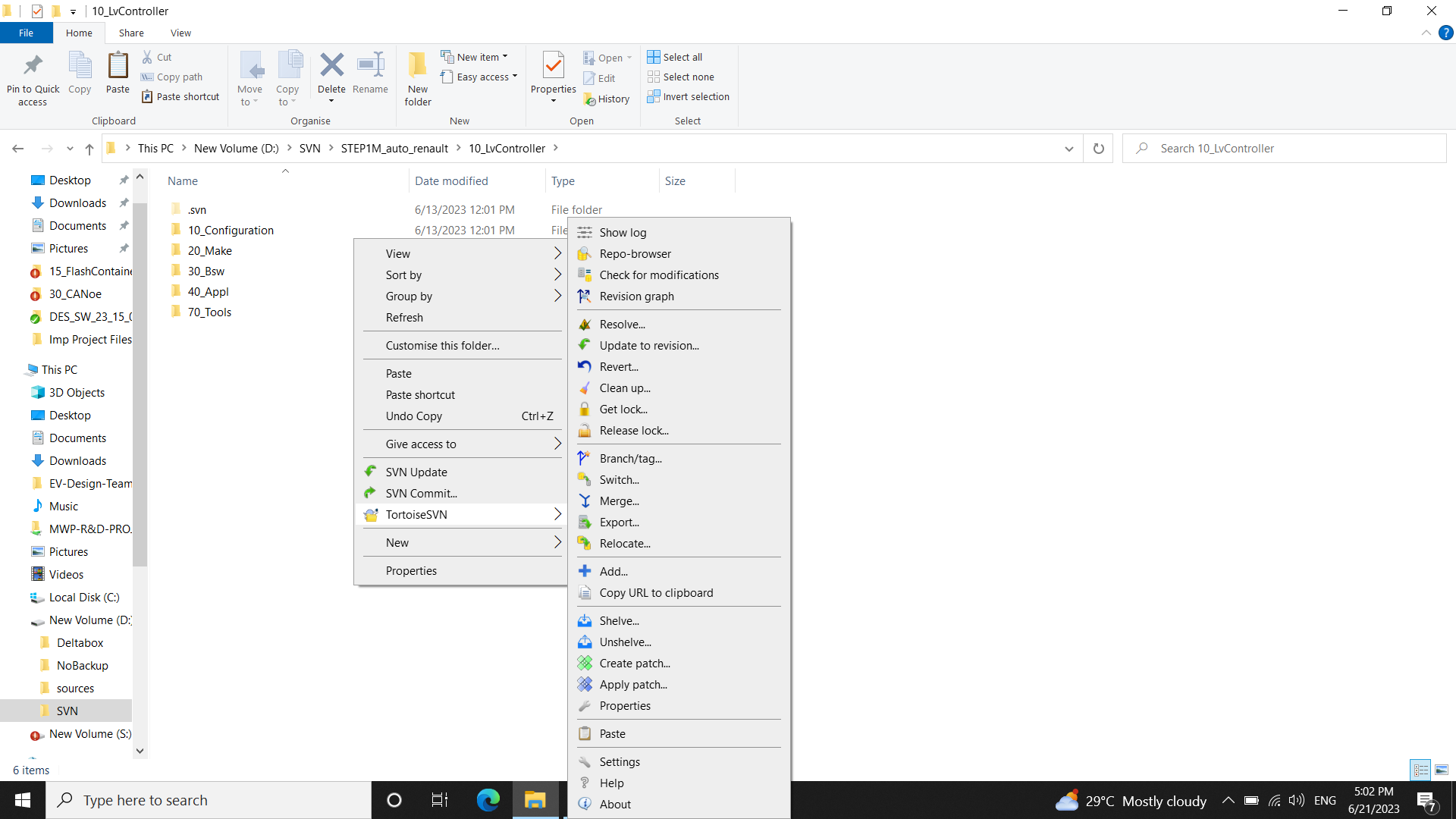
# 1. SVN Basics

## 1.1 Introduction

SVN stands for Subversion, which is a version control system used for managing and tracking changes to files and directories. It allows multiple developers to work on the same set of files concurrently and keeps track of all changes made to the files.

With SVN, developers can check out a copy of a project from a central repository onto their local machines. They can make modifications to the files, add new files, and remove existing files. Subversion keeps track of these changes and allows users to commit them back to the repository.

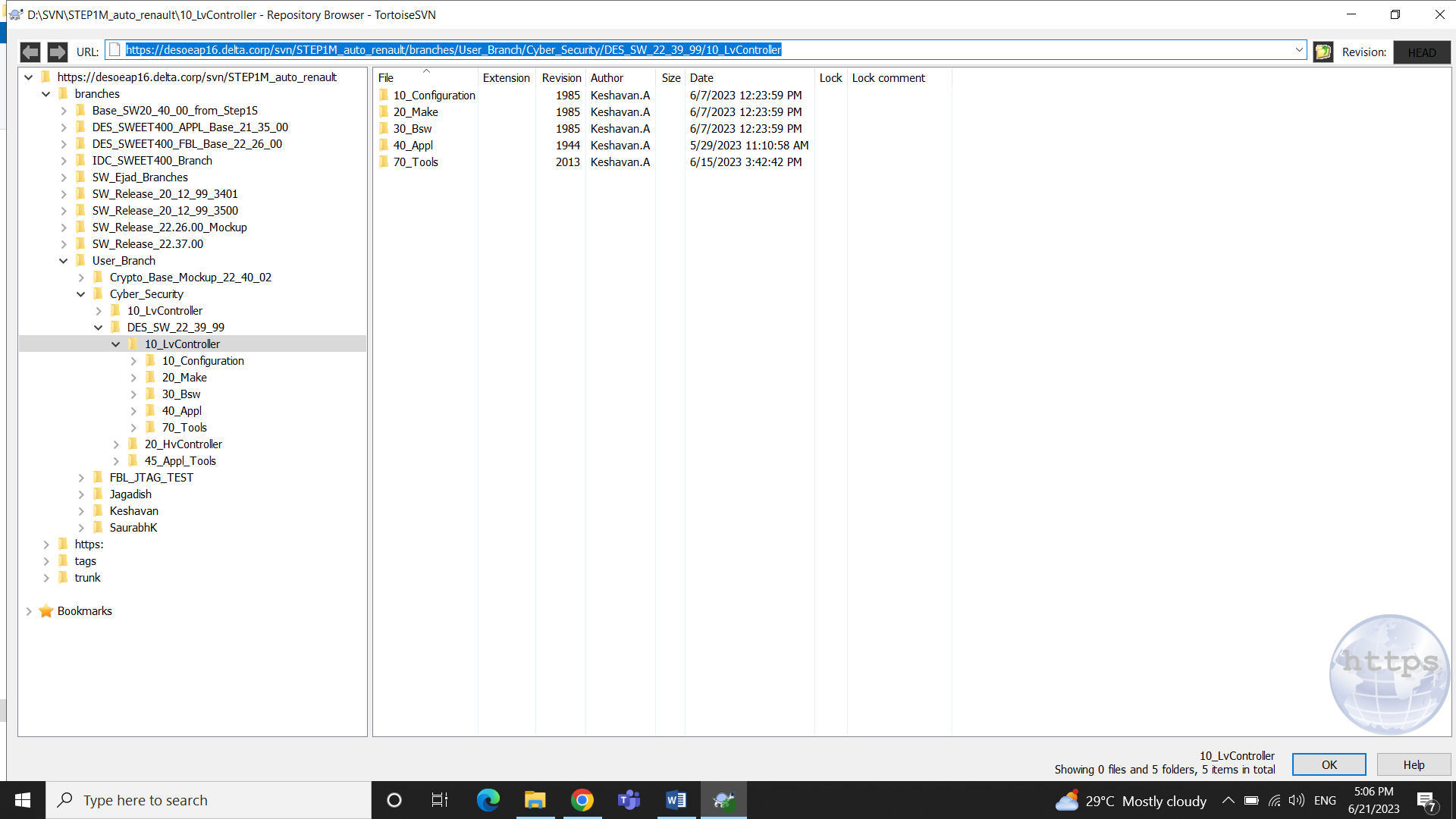
Tortoise SVN is the graphical client we used for SVN. Once you download the client we will be able to see the Tortoise SVN on right click and get its features.



## 1.2 Basic Uses

SVN Repository Browser

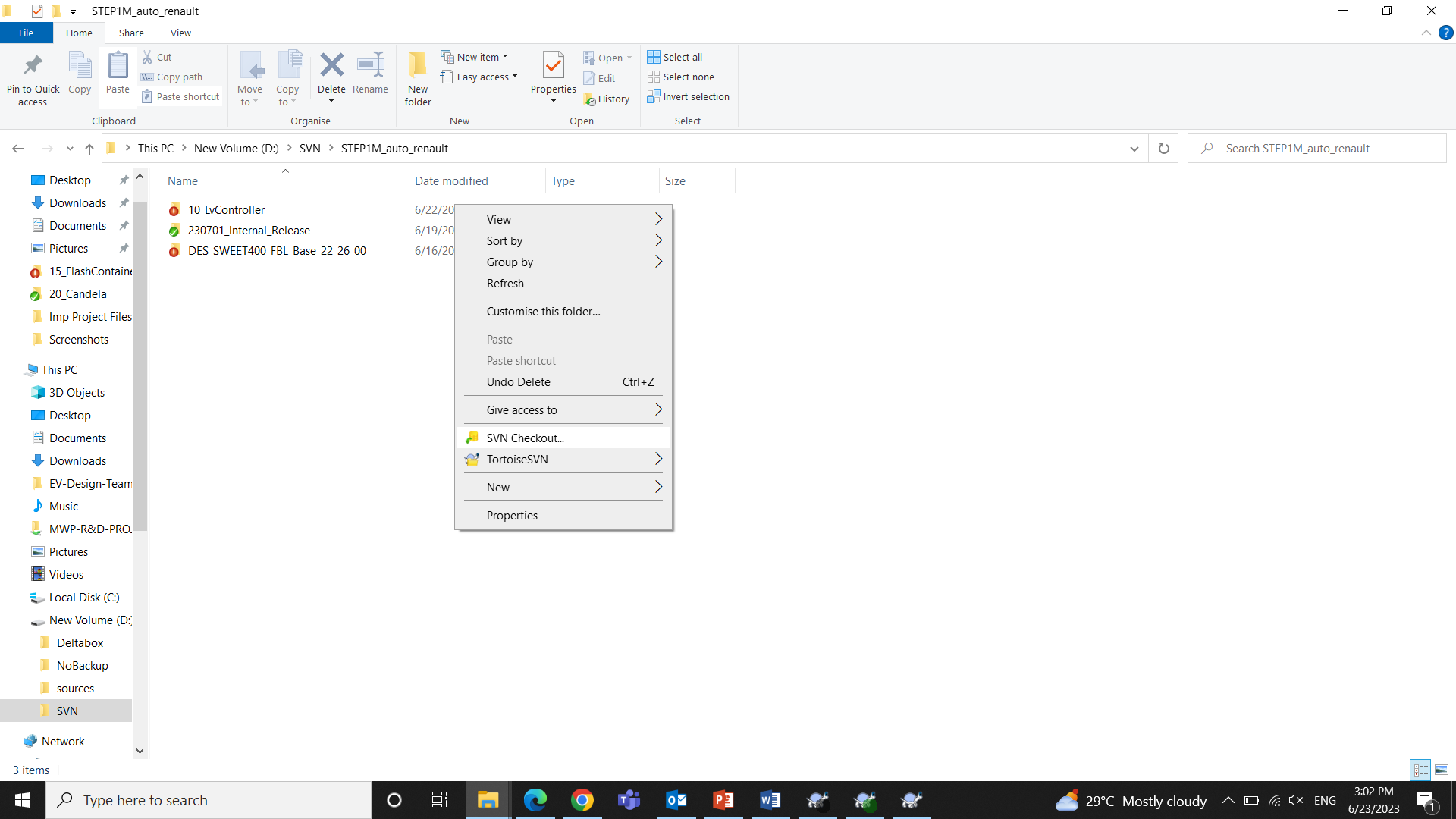
It allows users to browse and navigate through the directory structure of an SVN repository. It provides a hierarchical view of files and folders stored in the repository, giving users an overview of the project's structure.

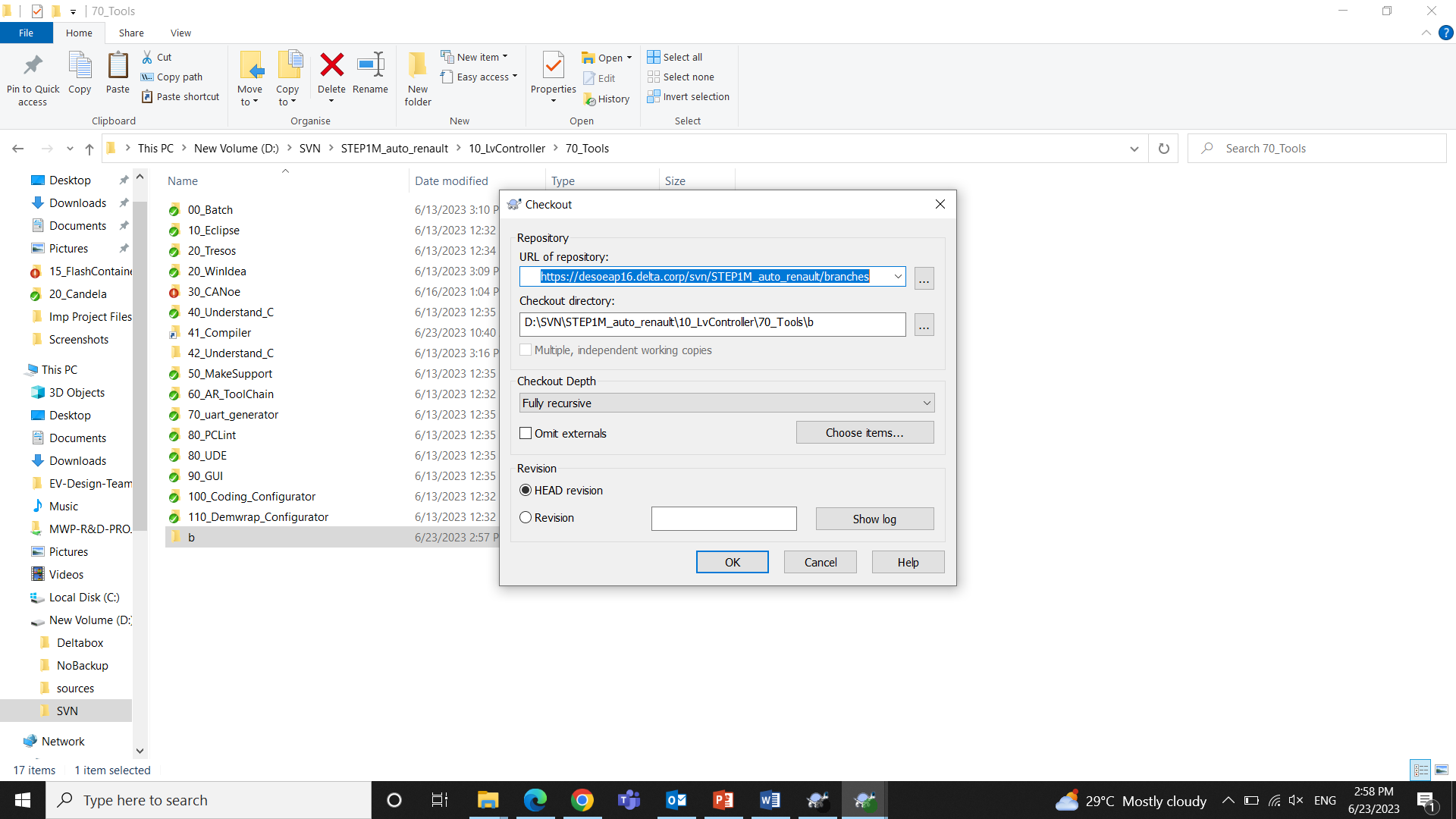


SVN Checkout

To Download the Repository into the local machine to get a working copy. In order to checkout we will open the SVN Repository Browser where we will type the URL.

Select **SVN Checkout** > Type in the URL and the path where it is to be downloaded.





SVN Update

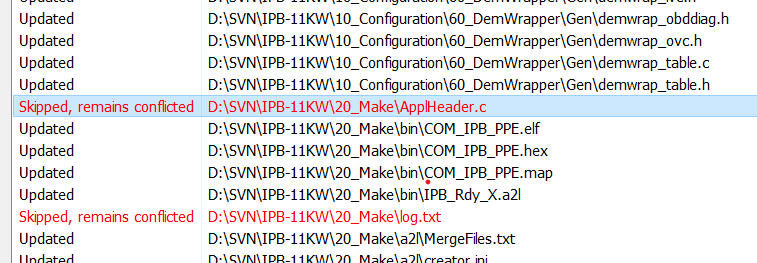
To Update the checked out Repository to the latest version.

Conflict

While Updating the SVN Repo, there may be cases of Conflicts.

Some files may be conflicted, which means that those files have a different version in our local repository. Such files may be edited by others who are co working in the Repo and committed to the server but we didn’t commit we made the changes locally.

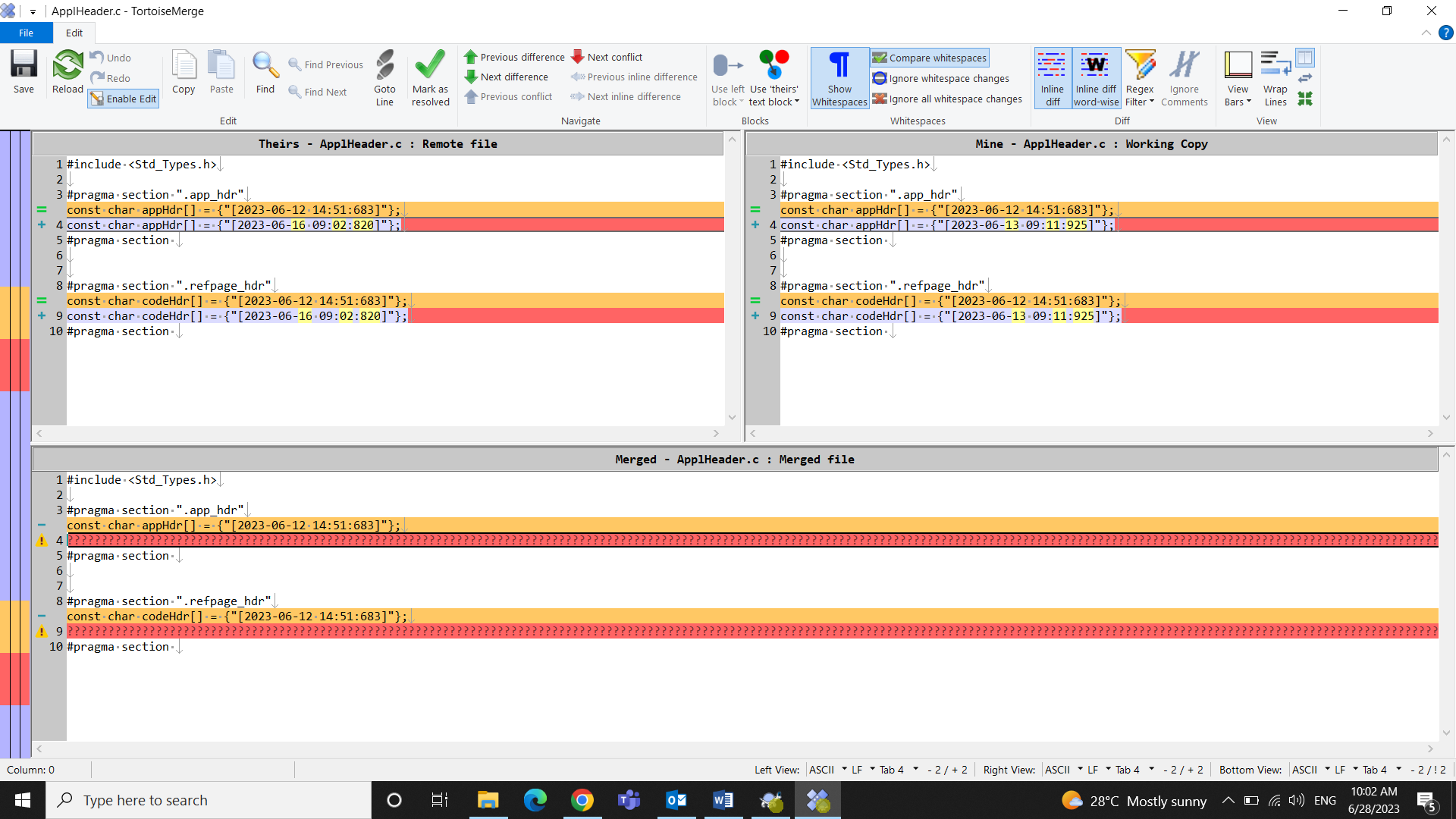
Conflict is actually in the working copy.

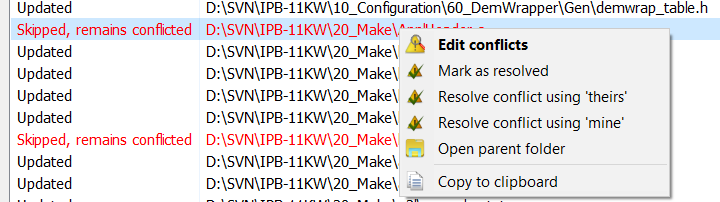


Resolving the Conflicts

When we right click on the conflict, we get options to edit and resolve the conflicts.

On clicking Edit Conflicts





**Mark as Resolved**: Ignoring the conflict.

**Resolve conflict using theirs**: Locally their update in the file will be preferred and my working copy will have those.

**Resolve conflict using mine**: Locally the changes made by me will be preferred and my working copy will have those, even though what is committed stays there in the server.

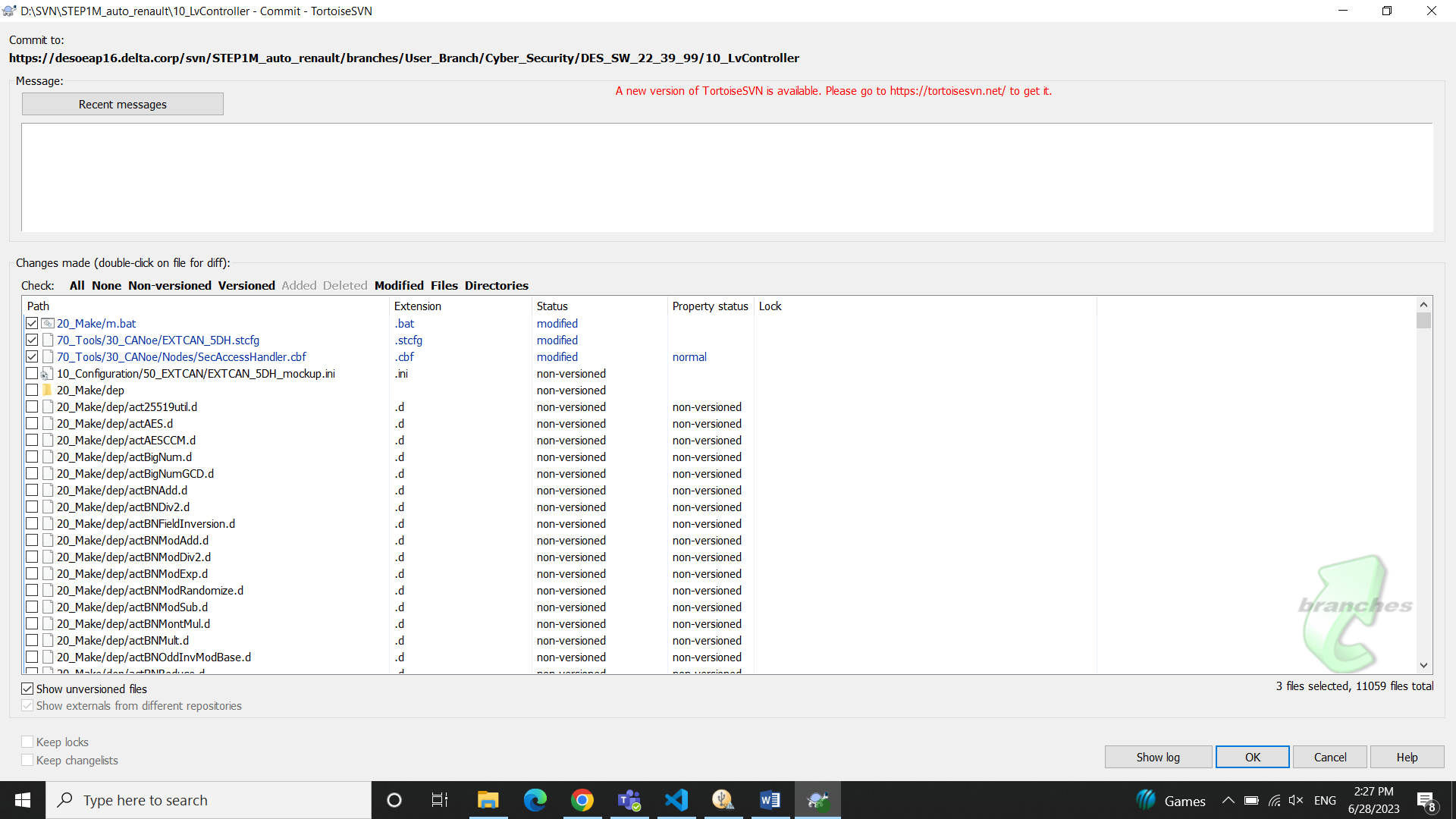
SVN Commit

To commit the changes did in working copy of the repository into the server.

When we move to the path where we made changes, then on selecting SVN commit, it will show the changes made in the files of that path.

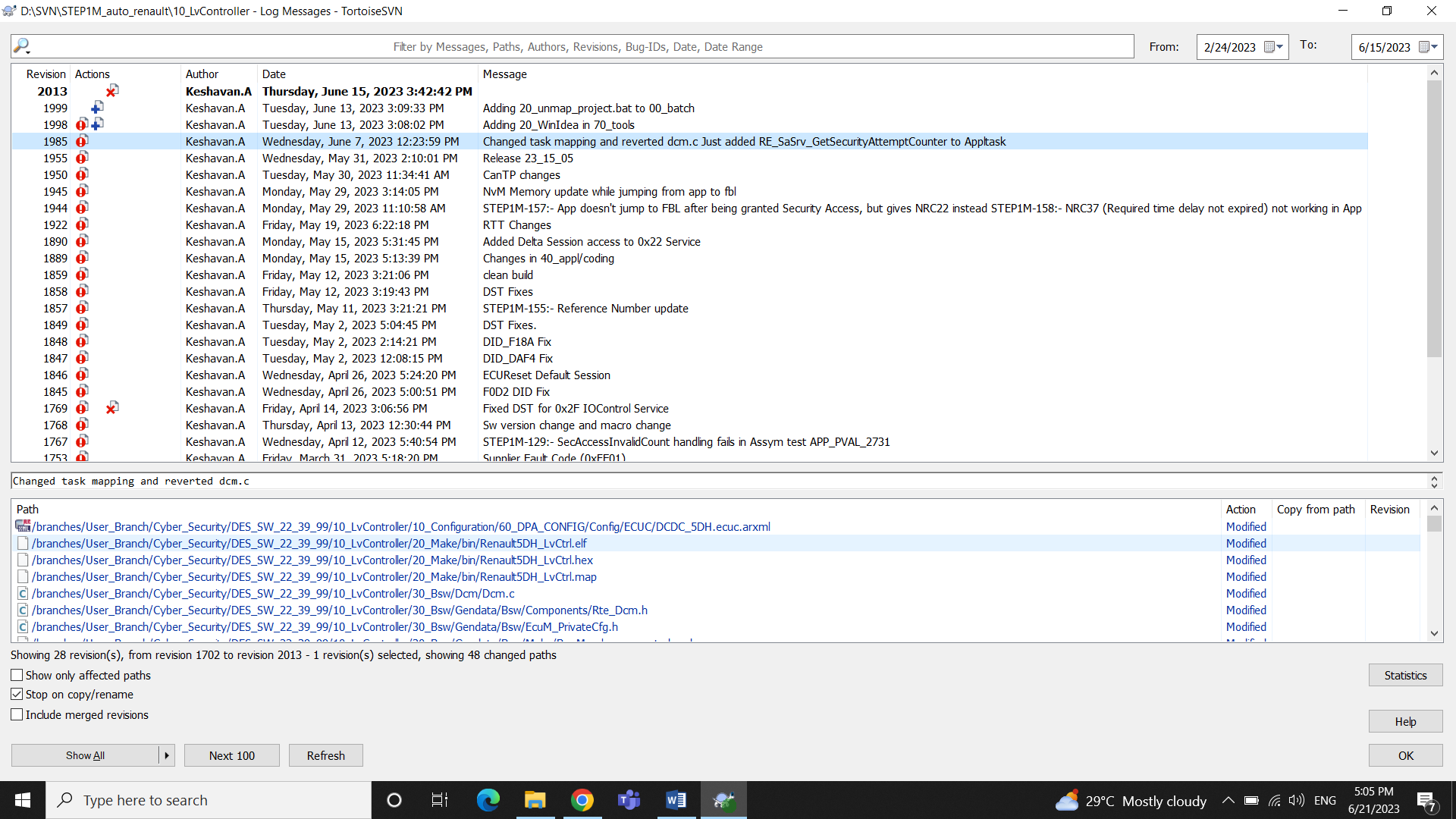
We don’t need to add (.d) files as they are dependency files, that doesn’t need to Commit.

Add the message and Click OK to commit the changes.

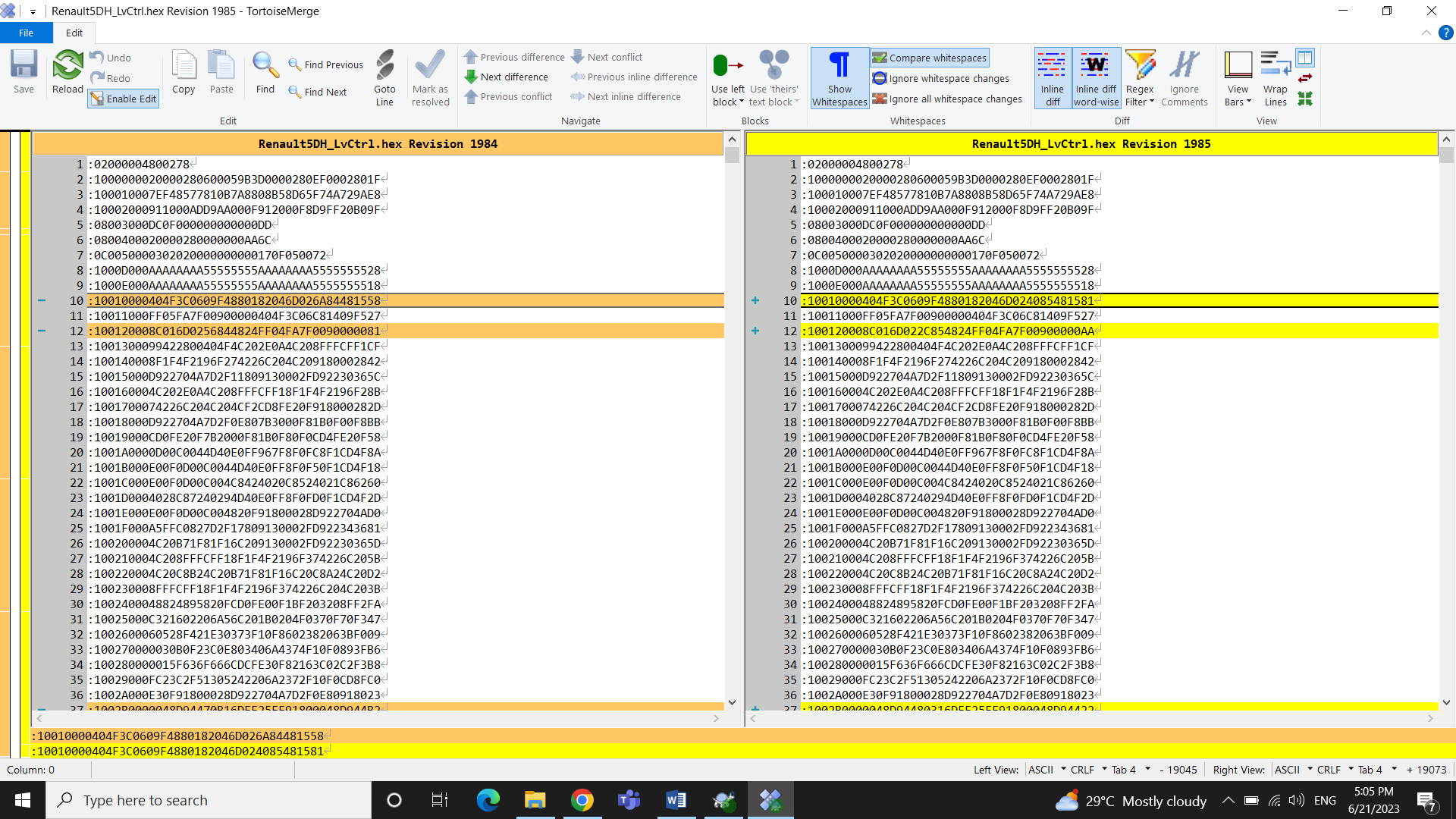


Show logs

To see the logs i.e. to see what all revisions happened in the repository. It contains the Revision No, Author who did the revision, time, date, messages. It also contains the path where changes happened in each revision we can compare the changes in each revision.



We can compare the changes on clicking the file in path console

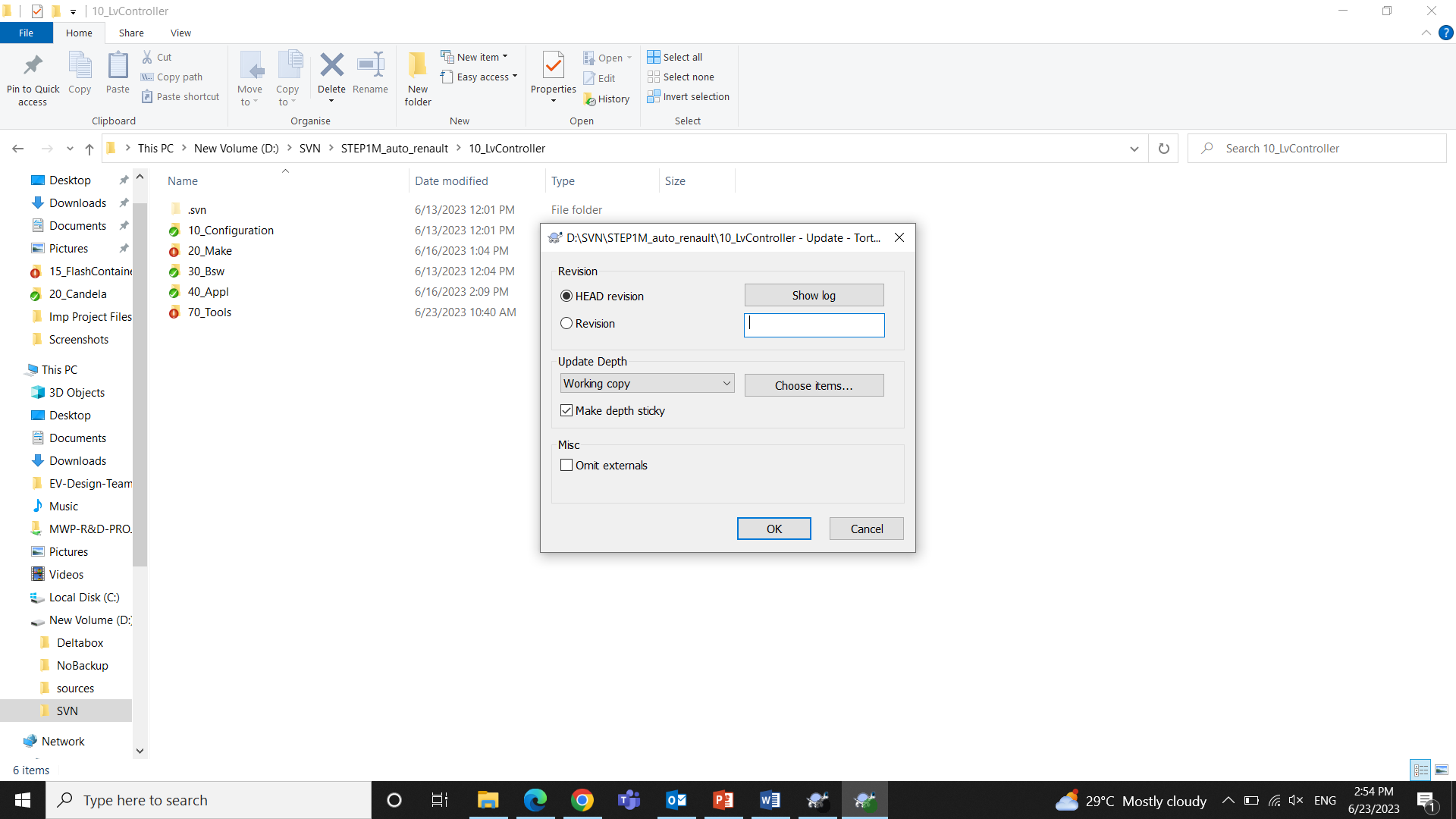


Check for Modifications

To identify the files and folders that have been modified in your working copy compared to the repository.

Update to Revision

We can move back to any previous revisions i.e. It allows you to roll back your working copy to a previous state and retrieve the files and directories as they existed at that revision. But make sure that any modifications made are backed up if you need them.

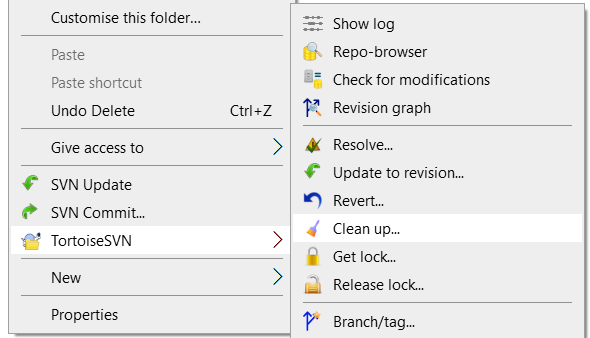


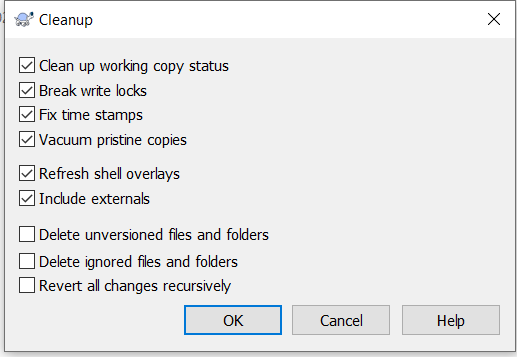
You can give the Revision number by looking to the logs.

Cleanup

While doing SVN Update, the update window sometimes asks you to cleanup.

Cleanup is used to resolve issues related to your working copy.





Make sure you don’t check the **delete options** unless you have a backup

Revert

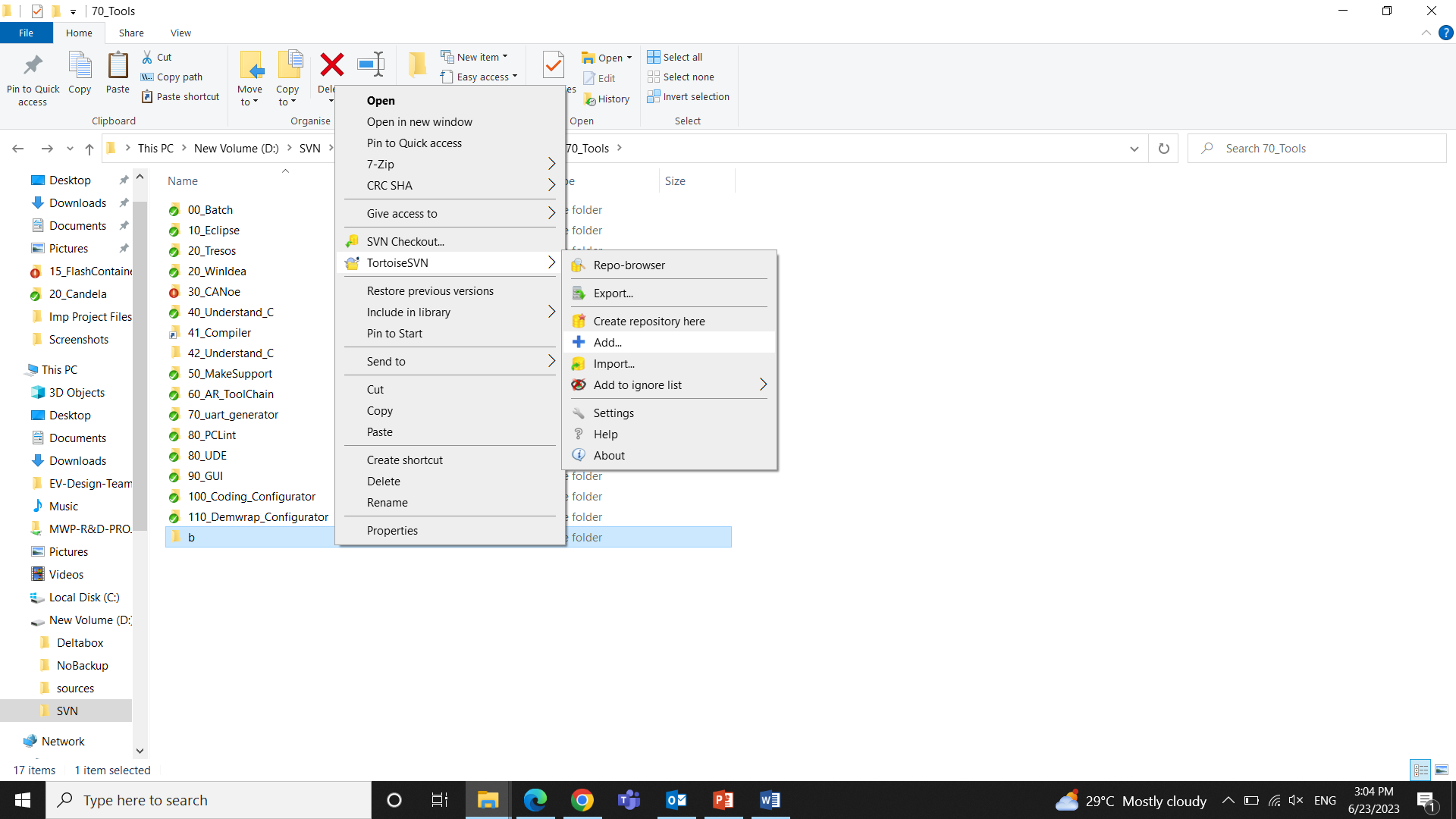
To undo all the local changes done in the working copy.

Get Lock

To lock while making changes in the trunk so that no other can interfere parallel.

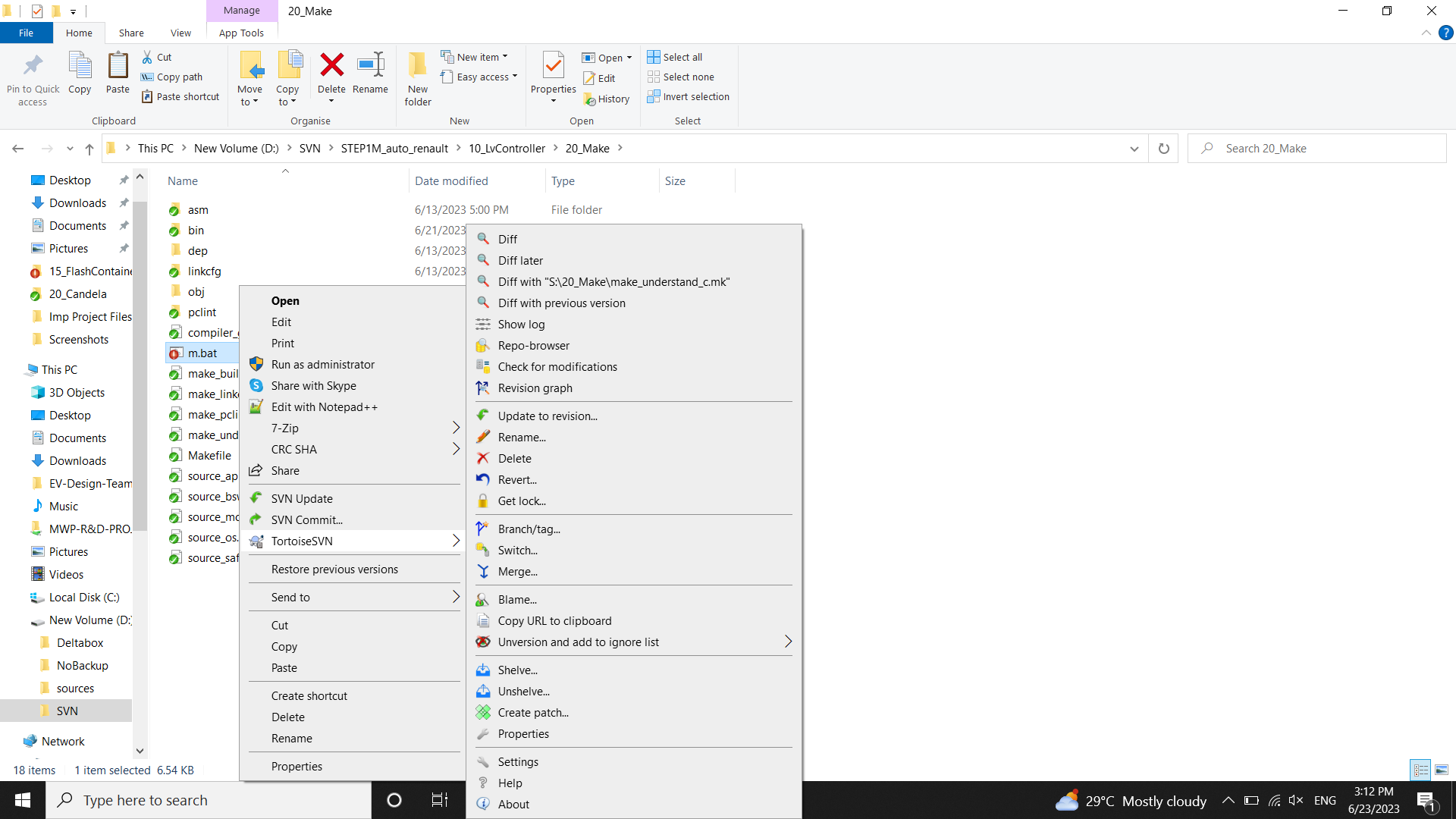
Add

To add any new files or folders into the Repository



Diff

If we made any changes to a file and then **right click** to open SVN. It will show **diff** from where we can compare that file with previous revisions.



**CHAPTER 2**

# 2. Building , Flashing and Debugging

Building a file refers to compiling the Source file by linking all the object and linker files using the specified compiler. The process gives an executable file which can be flashed onto ECU.

## 2.1Building

We need to create S drive which is the copy of project files in D:\SVN\STEP1M\_auto\_renault\10\_LvController

We will be making changes related to project in S drive which will be affected in D drive too.

Why to Create S drive?

In order to standardize the path of files as many developers are involved in the project. So we use a common drive S as many files involving stating the path.

Step 1: To Create S drive

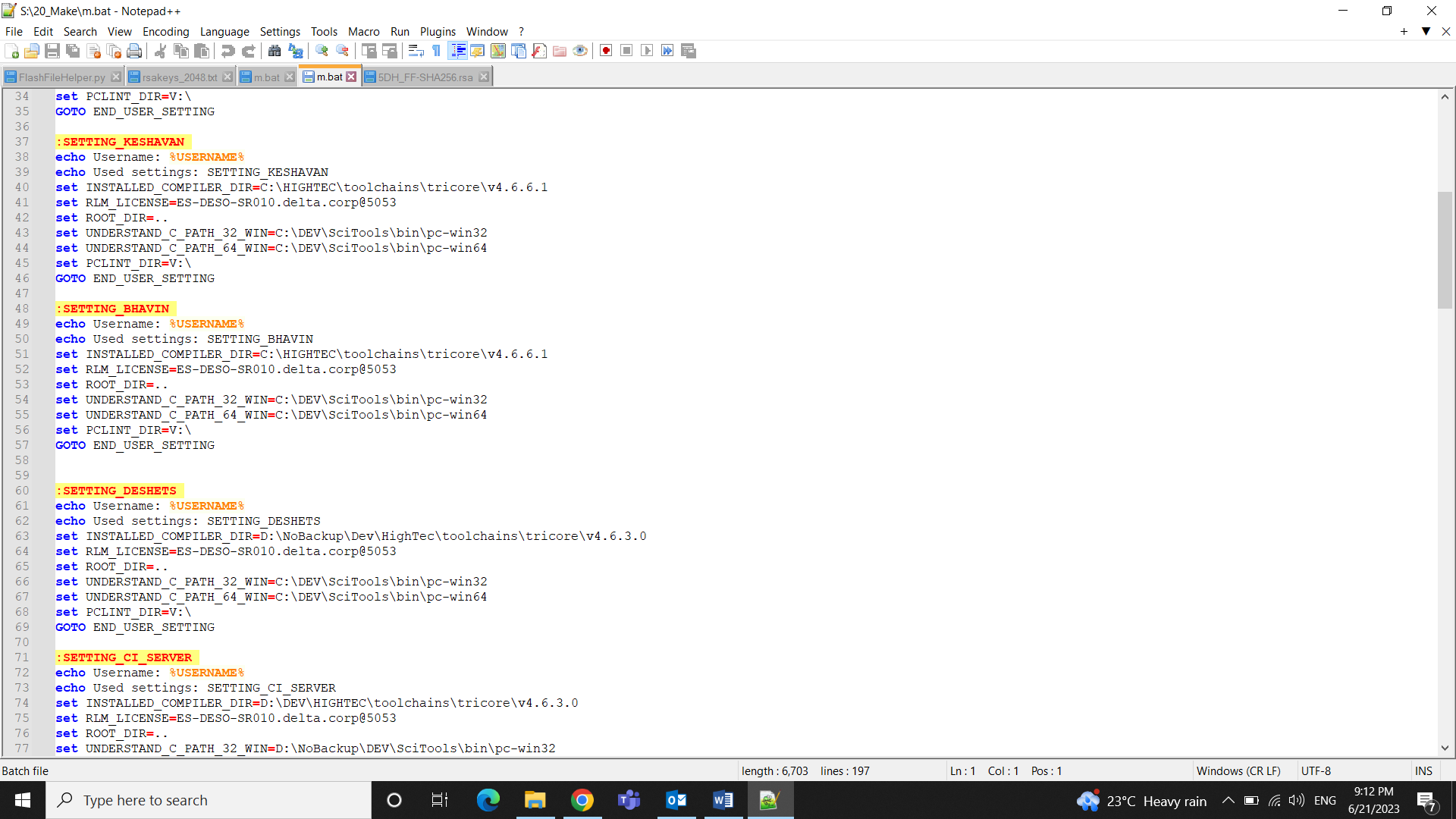
D:\SVN\STEP1M\_auto\_renault\10\_LvController\70\_Tools\00\_Batch

In this path open Batch file named 10\_map\_project



Step 2: S:\20\_Make

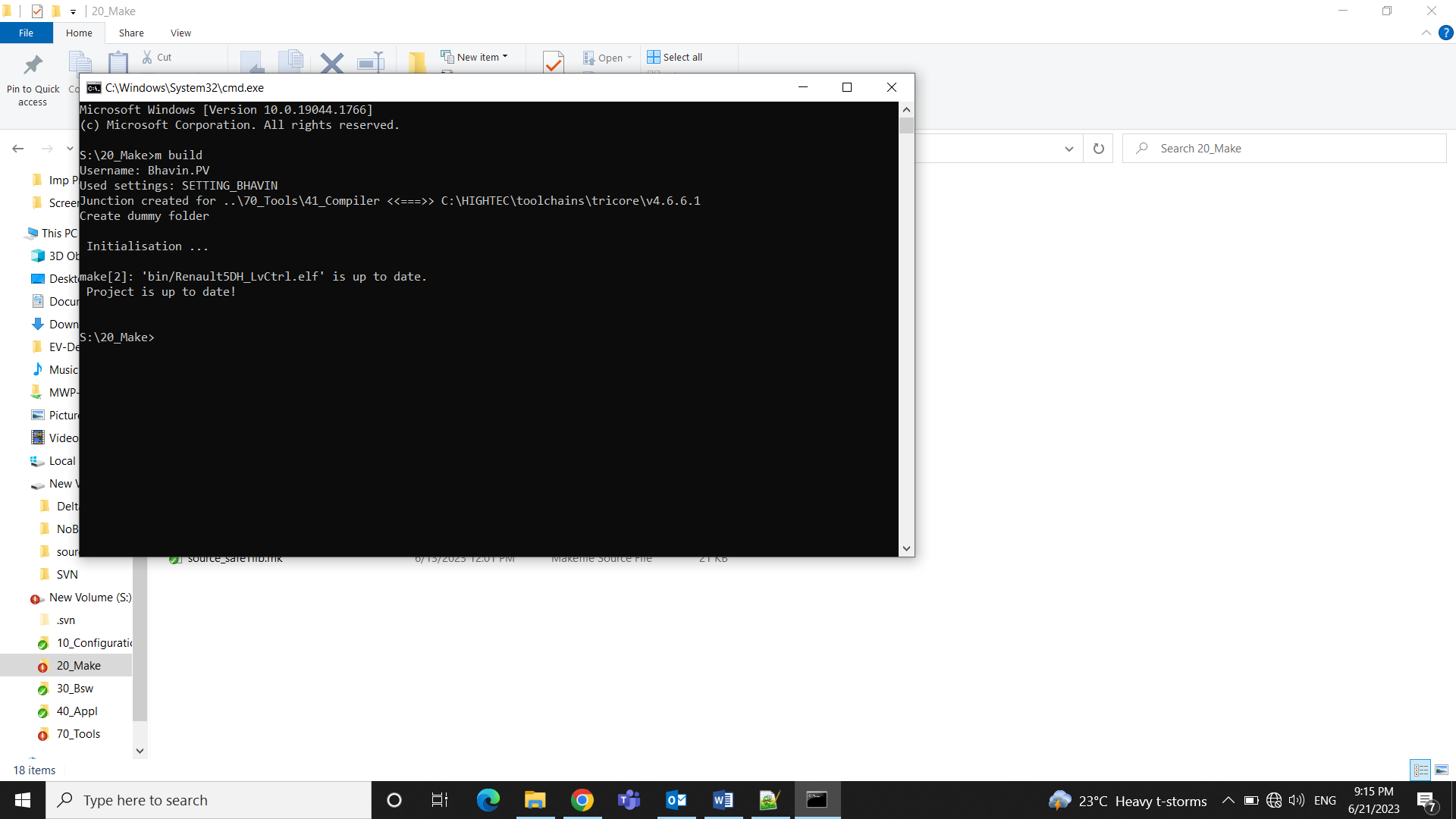
In the above path open m.bat file using Notepad++ and add the username and path for the compiler.



Step 3: Delete the current files (.elf , .hex, .map) from bin in the path S:\20\_Make

Step 4: Type **cmd** in S:\20\_Make to open the command prompt

Step 5: Type **m build** to do the incremental build, **m rebuild** to do complete fresh build



2.2 Flashing

Flashing code into an Electronic Control Unit (ECU) refers to the process of writing or updating the software that controls the functions and behavior of the ECU.

### 2.2.1 Flashing the Application

Connections:

Winidea Debugger is connected to ECU through JTAG and to computer through USB

CAN is connected to computer through USB and to ECU through connector.

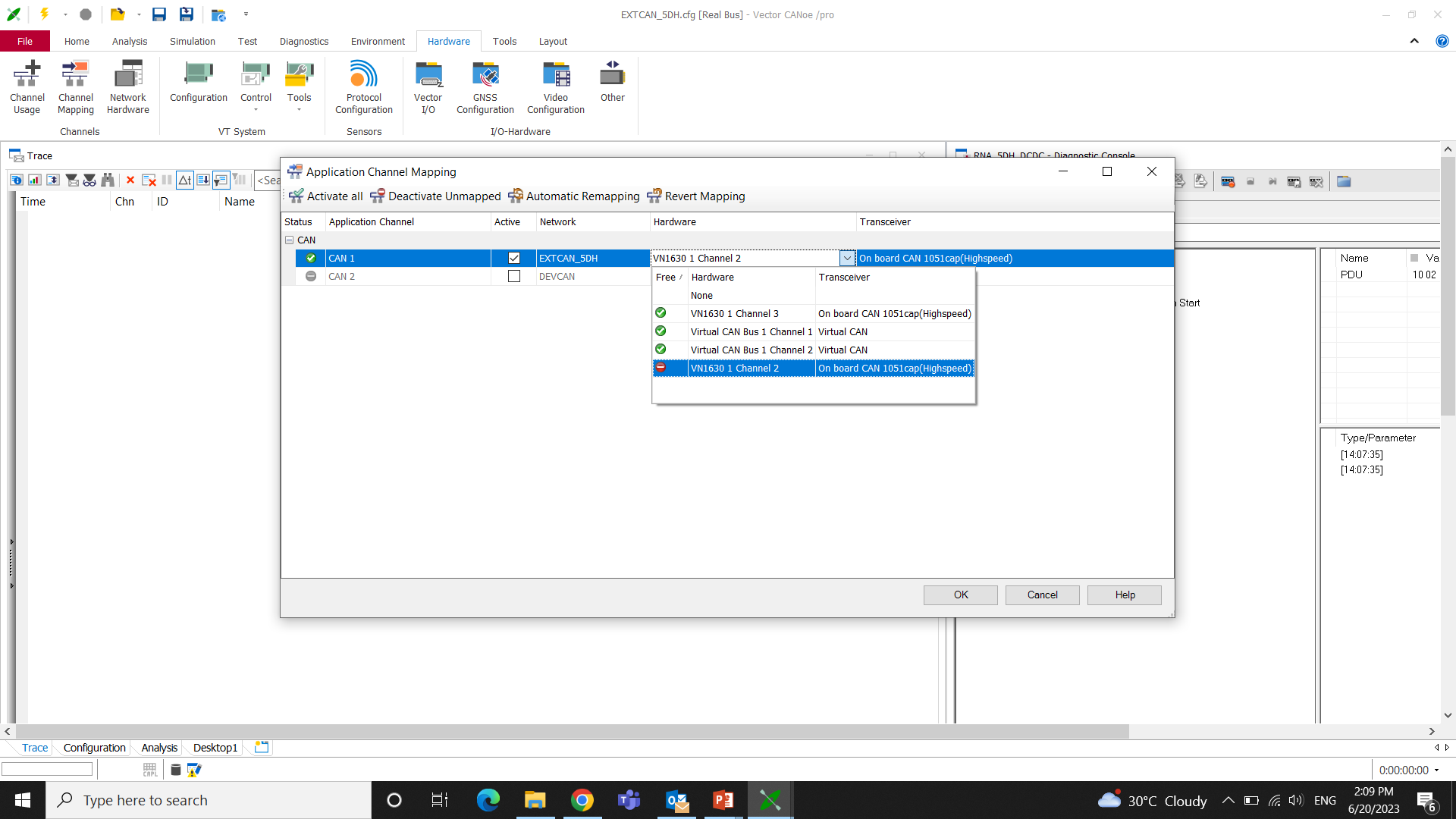
Software used: Winidea debugger, CANoe tool

Step 1: S:\70\_Tools\30\_CANoe

In the above path open EXTCAN\_5DH.cfg using CANoe

Step 2: In **Hardware** Open **Channel Mapping** in the CANoe Software

Check correct channel is mentioned in On Board CAN channel by confirming it from the CAN tool i.e. through which channel ECU is connected to CAN



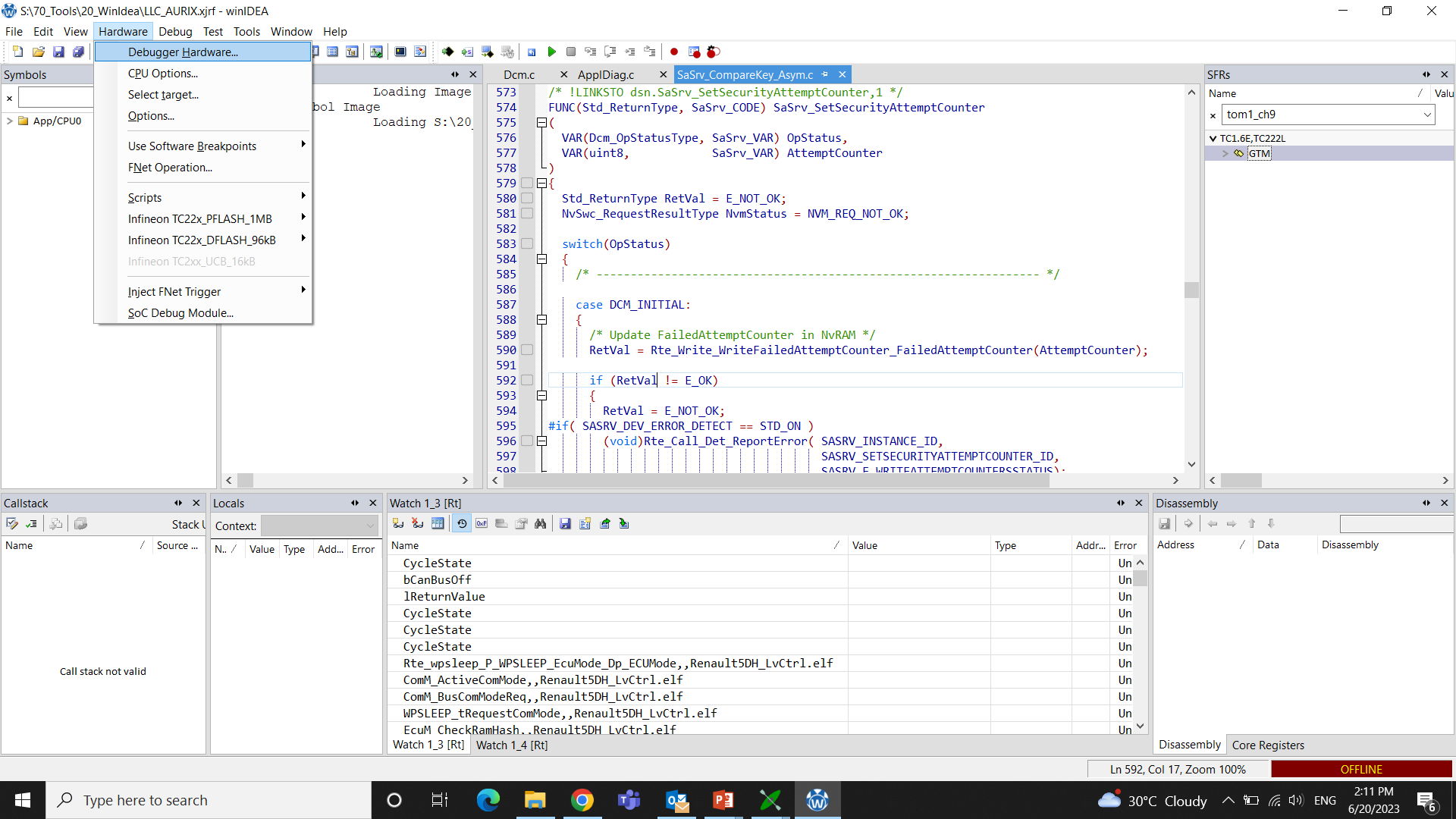
Step 3: S:\70\_Tools\20\_WinIdea

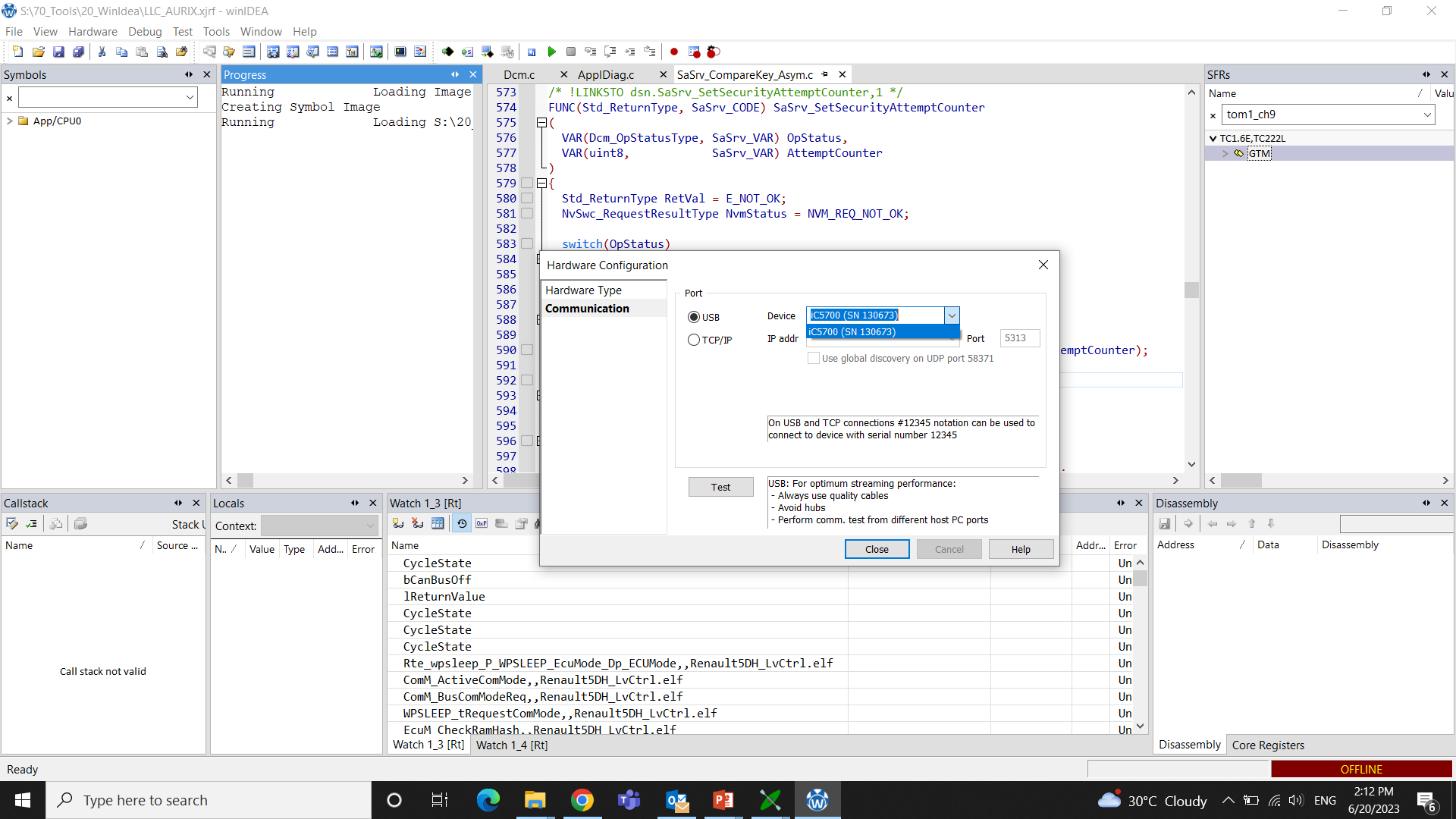
In the above path open LLC\_AURIX.xjrf using Winidea software

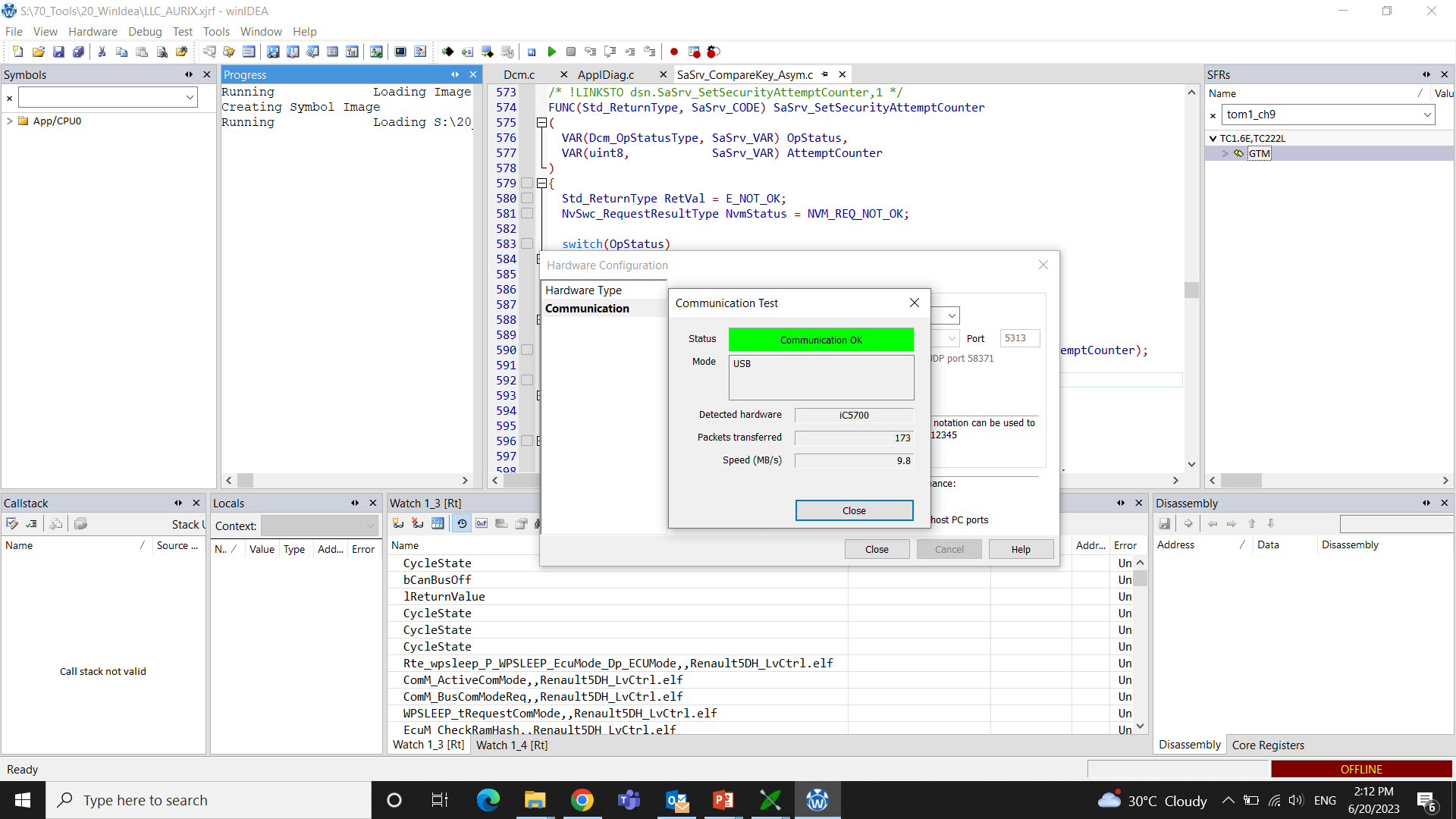
Step 4: To check communication between ECU and debugger

In Windiea software select **Hardware** > **Debugger Hardware** **> Communication**>**Test**

It will show “**Communication OK** “if the communication is properly established between both ECU and debugger.

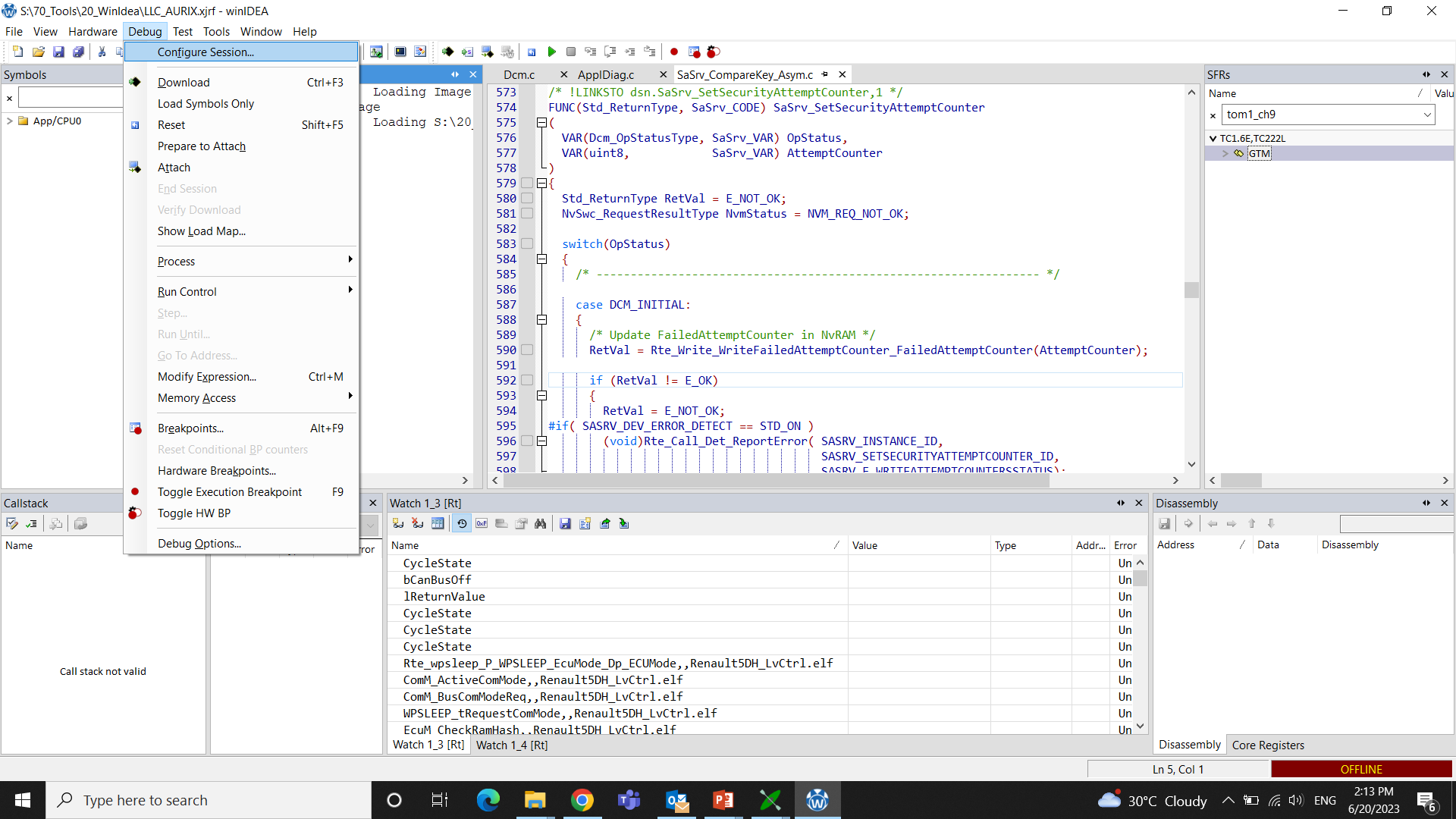


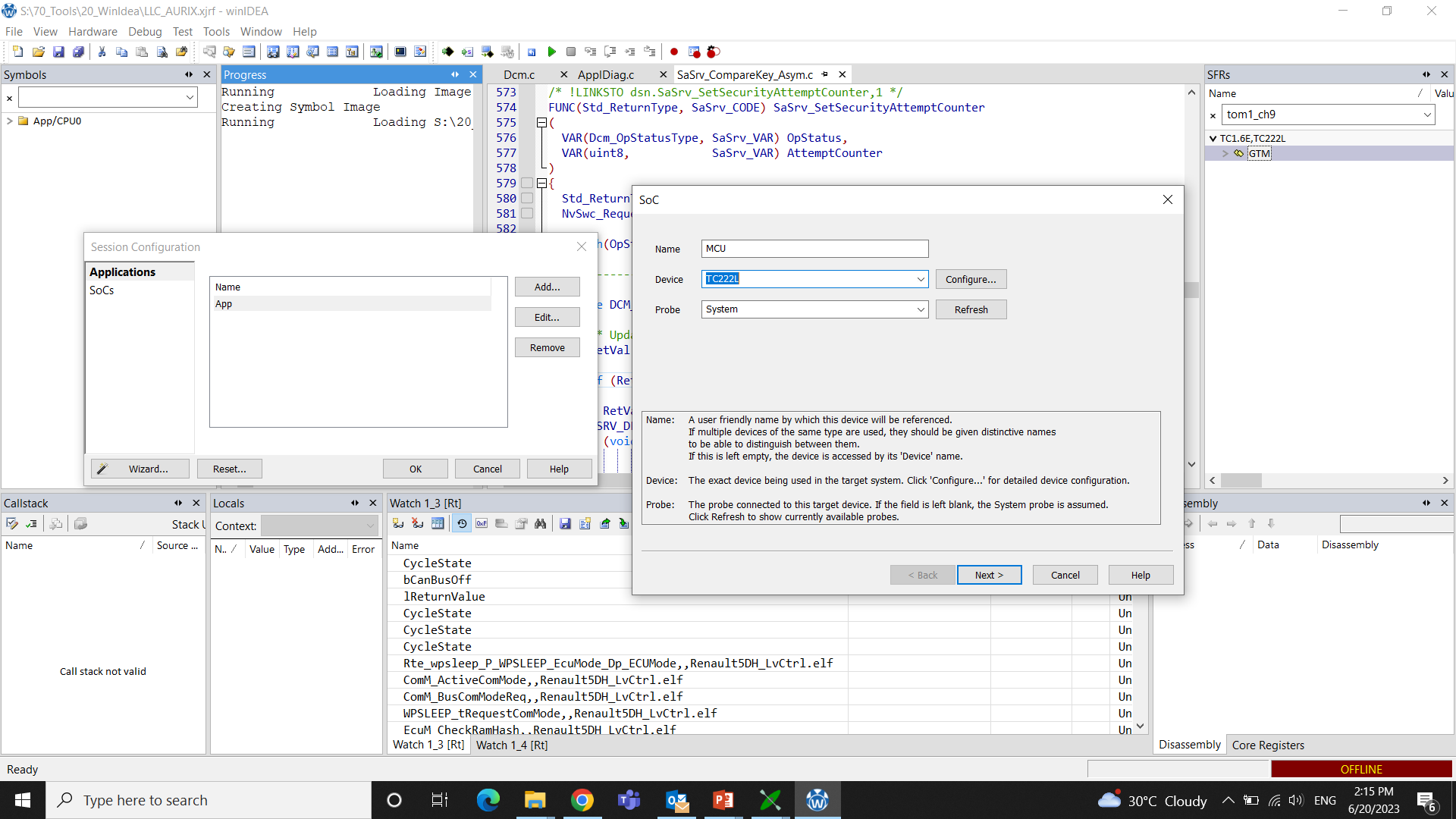


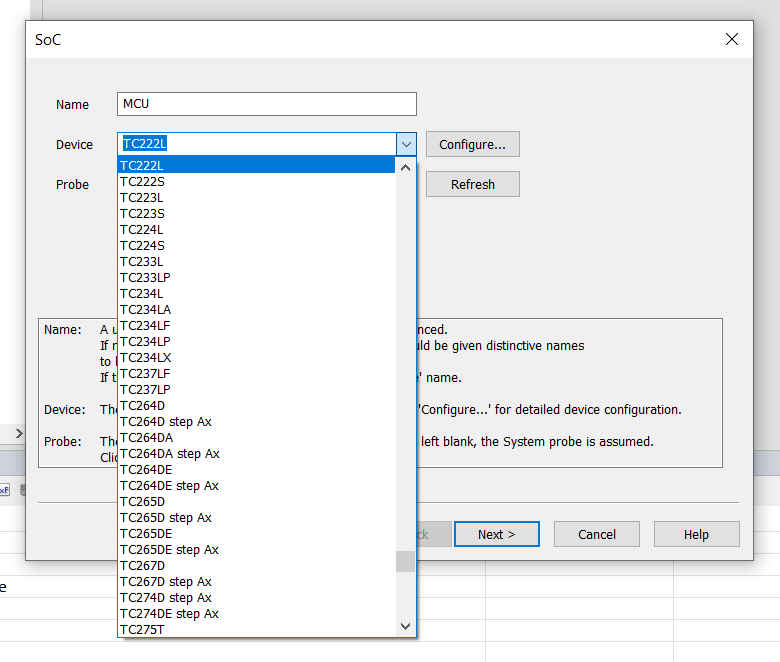


Step 5: In Winidea software select **Debug** > **Config Session** >**Wizard**

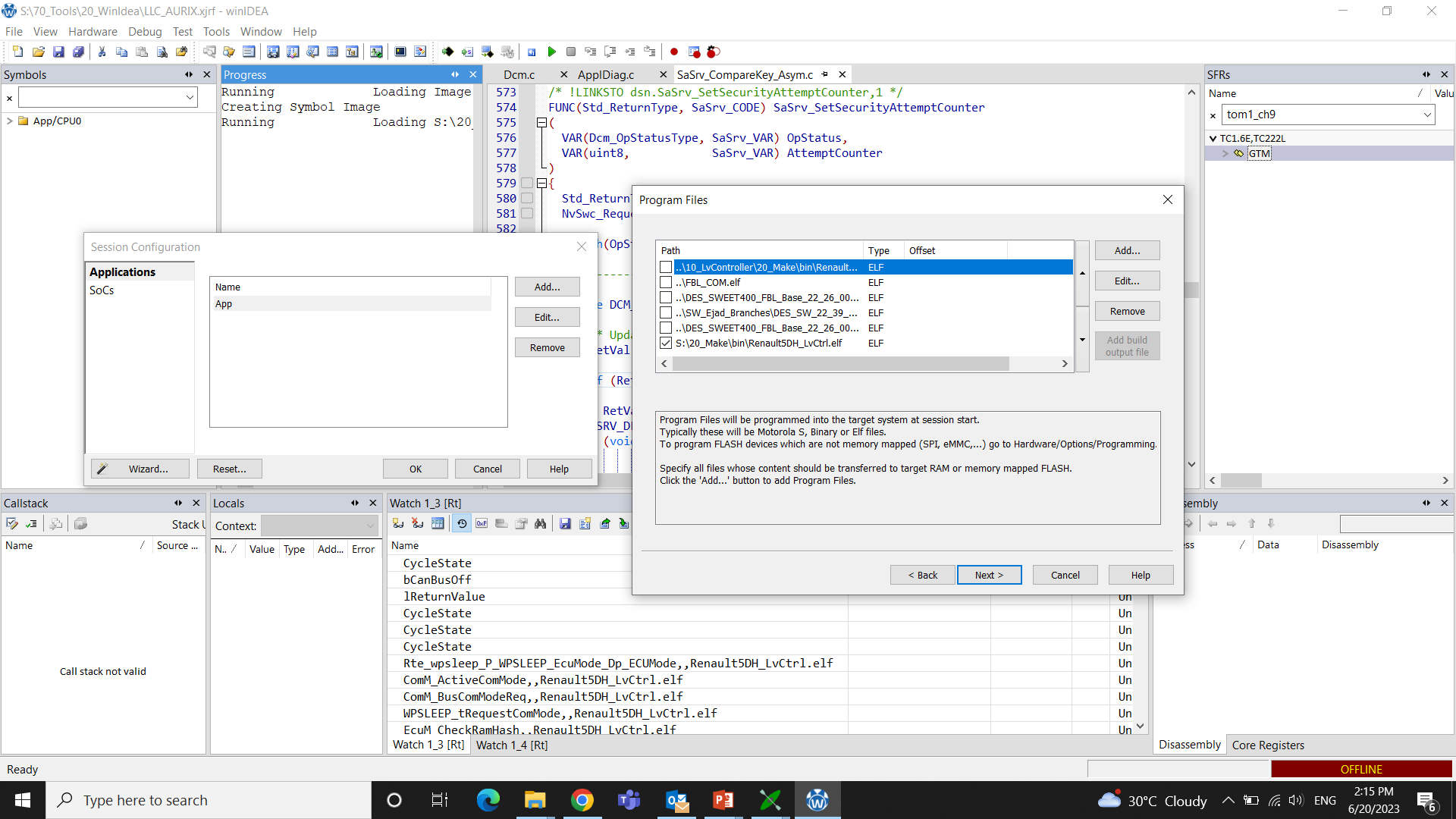
Add proper. elf file from S:\20\_Make\bin







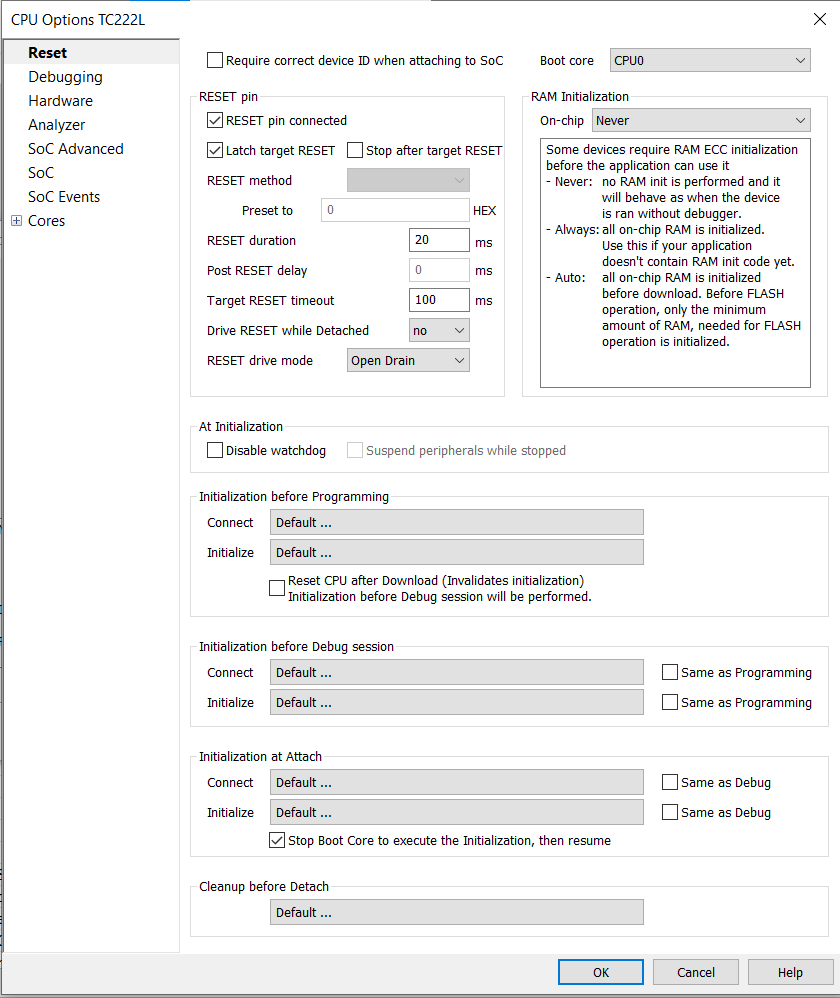
Also make sure that Correct Device is mentioned in the Device section. Here we are using **TC222L** SOC from Infineon. The Probe section is Mentioned as **System**.

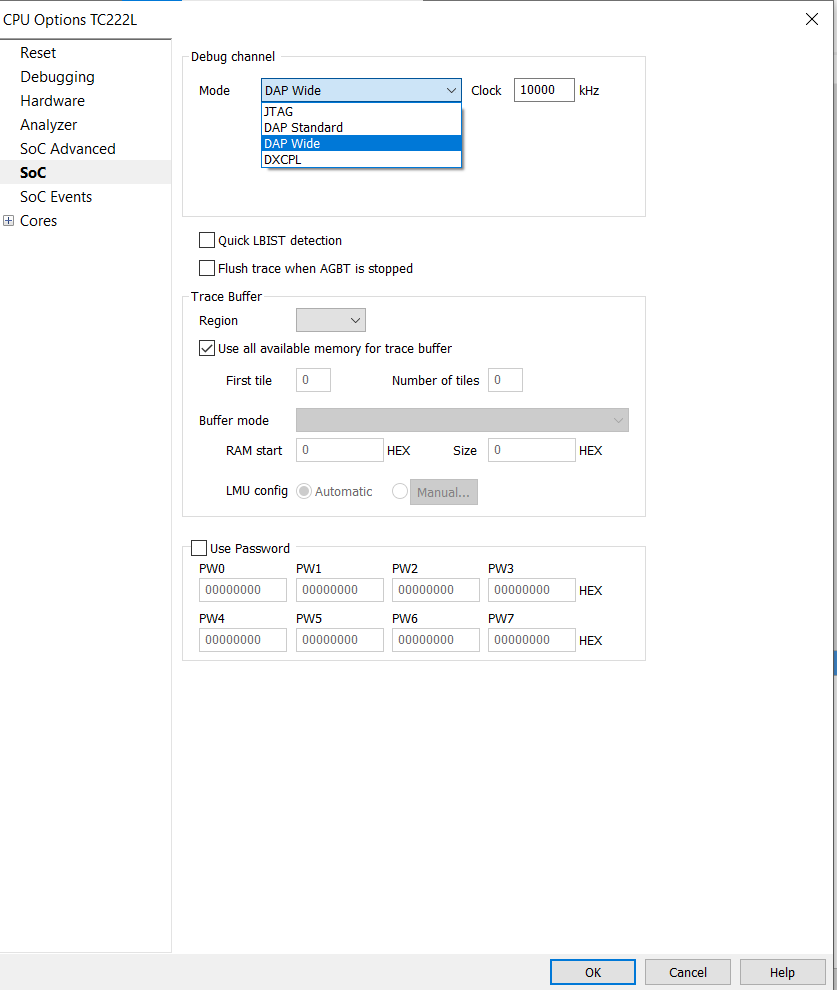


Step 6: In **Wizard** Select **Configure**

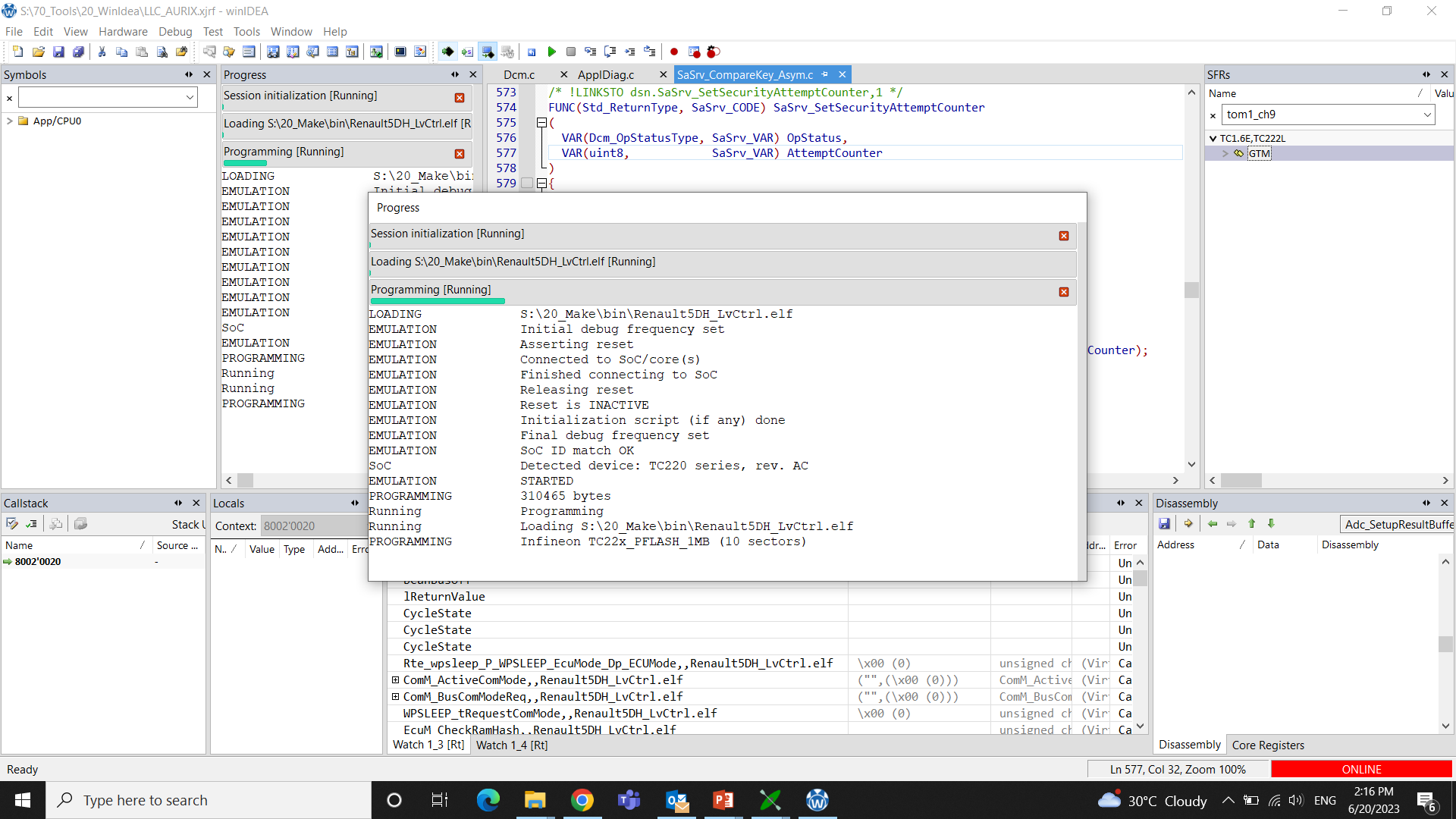
Reset, Debugging, Hardware, Analyzer, SoC advanced, SoC events, Cores Options are kept as Default

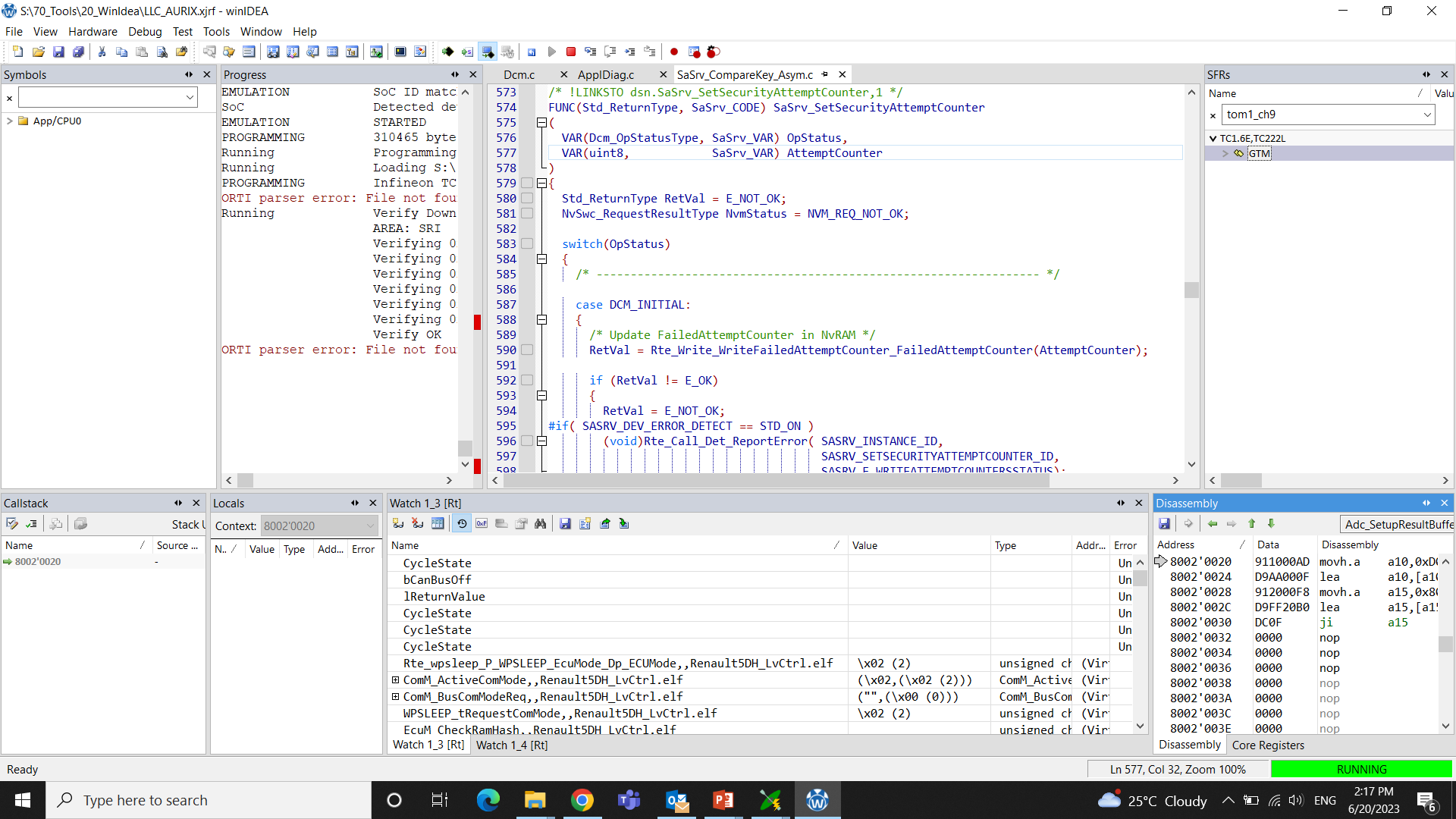
In **SoC** Option **Debug Channel Mode** is Kept as “**DAP WIDE**” and Clock Frequency as **10000Khz**





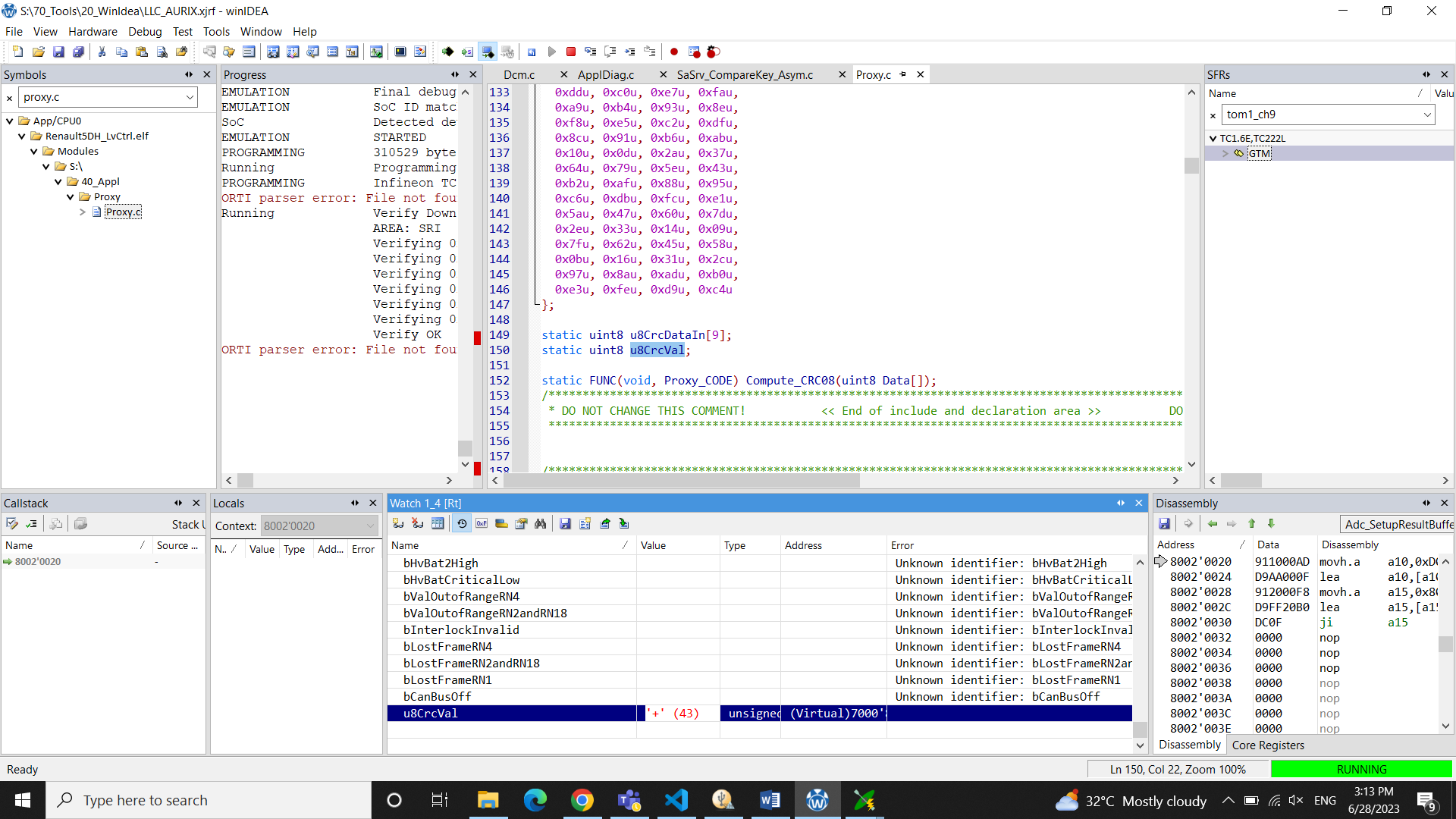
Step 7: In Winidea select **Download** and **Play**

****

****

To add alive counter, that gives the state of variables

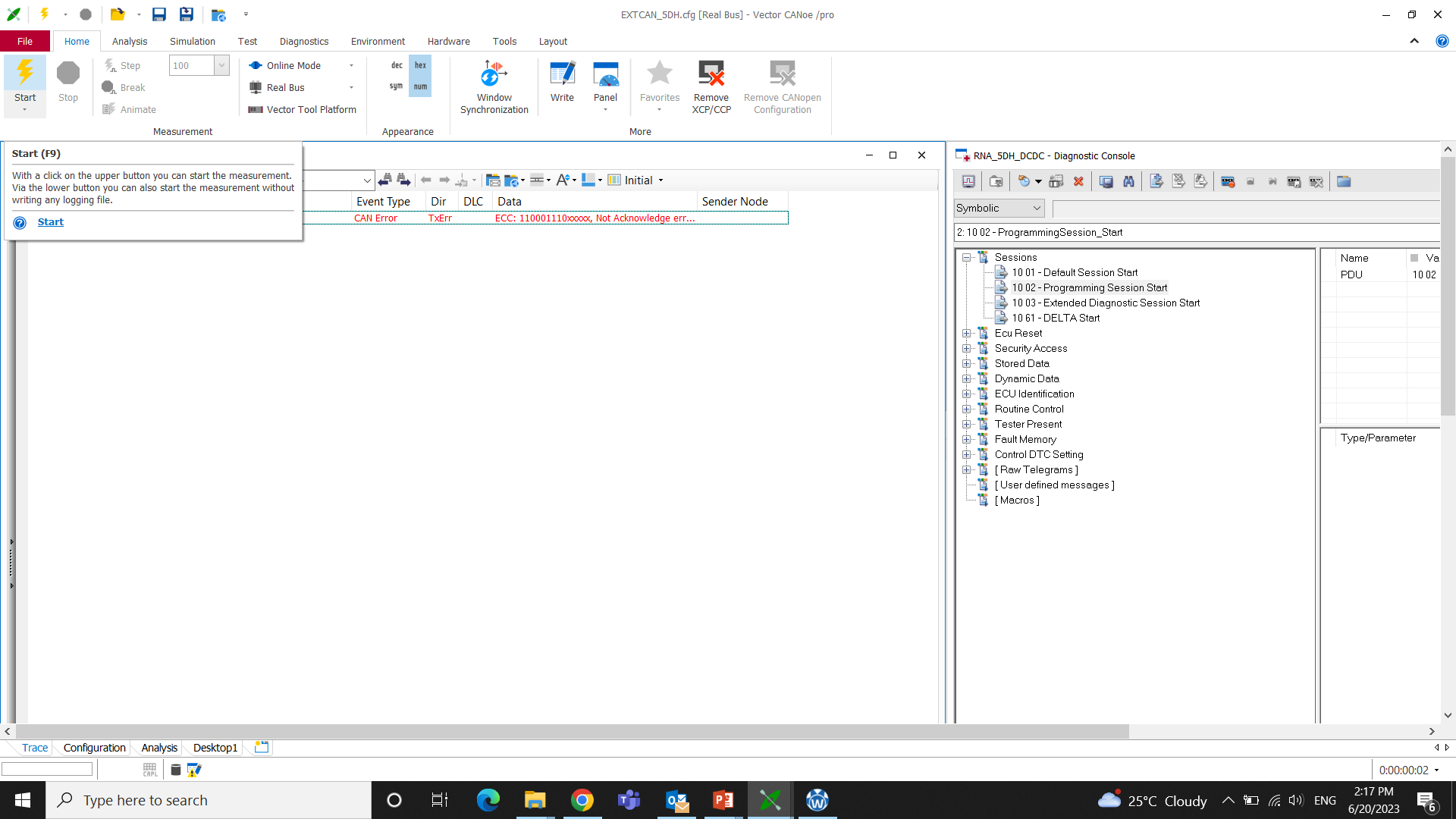
Copy the variable name and then paste it to the watch window.

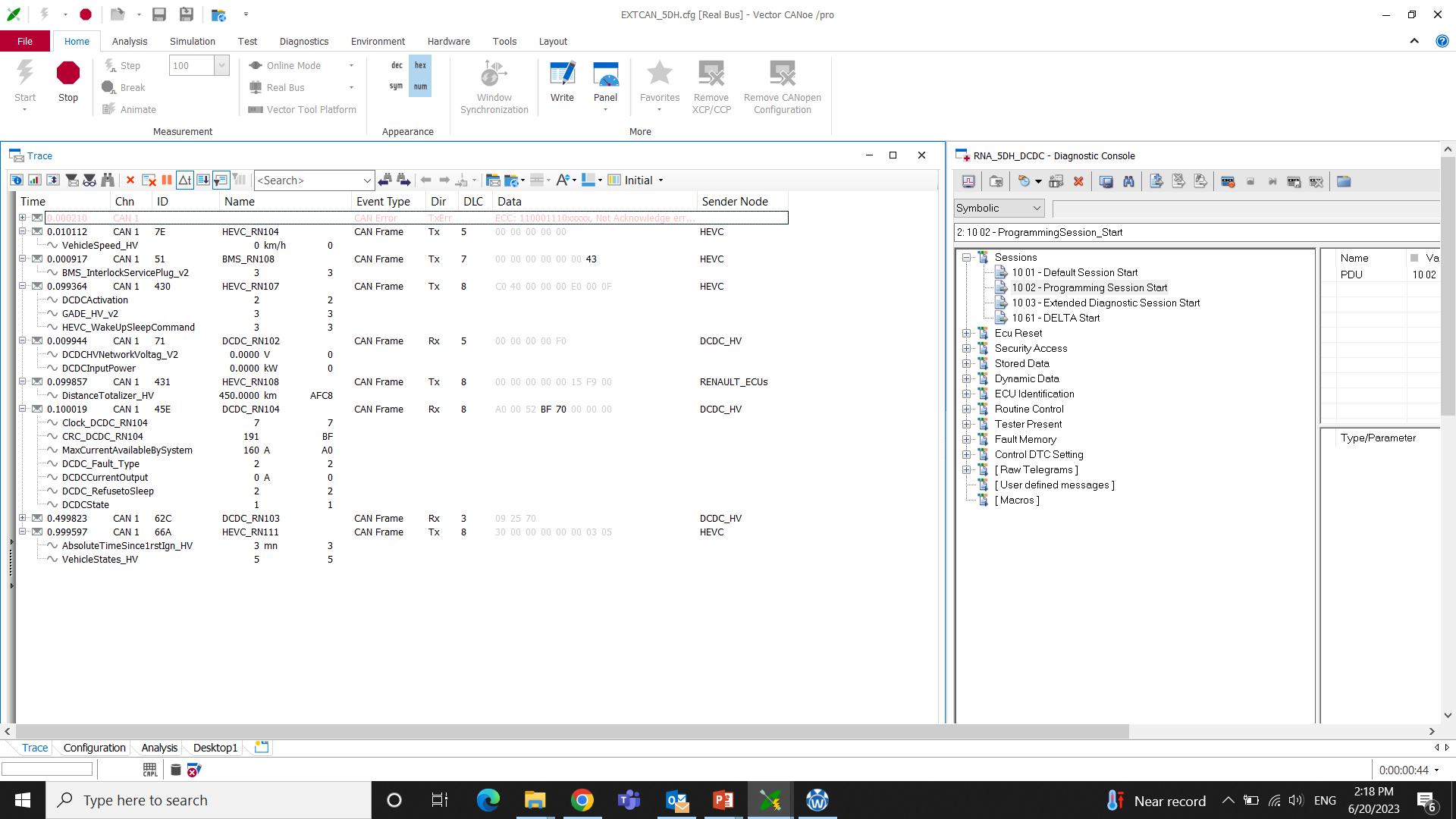


Here we need to watch a variable from Proxy.c file in 40\_Appl

We can only monitor global variables in the watch window

Step 8: Select **Start** in CANoe





### START CANOE, FLASH IN WINIDEA ,POWER CYCLE.

### 2.2.2 Flashing the Bootloader

Connections:

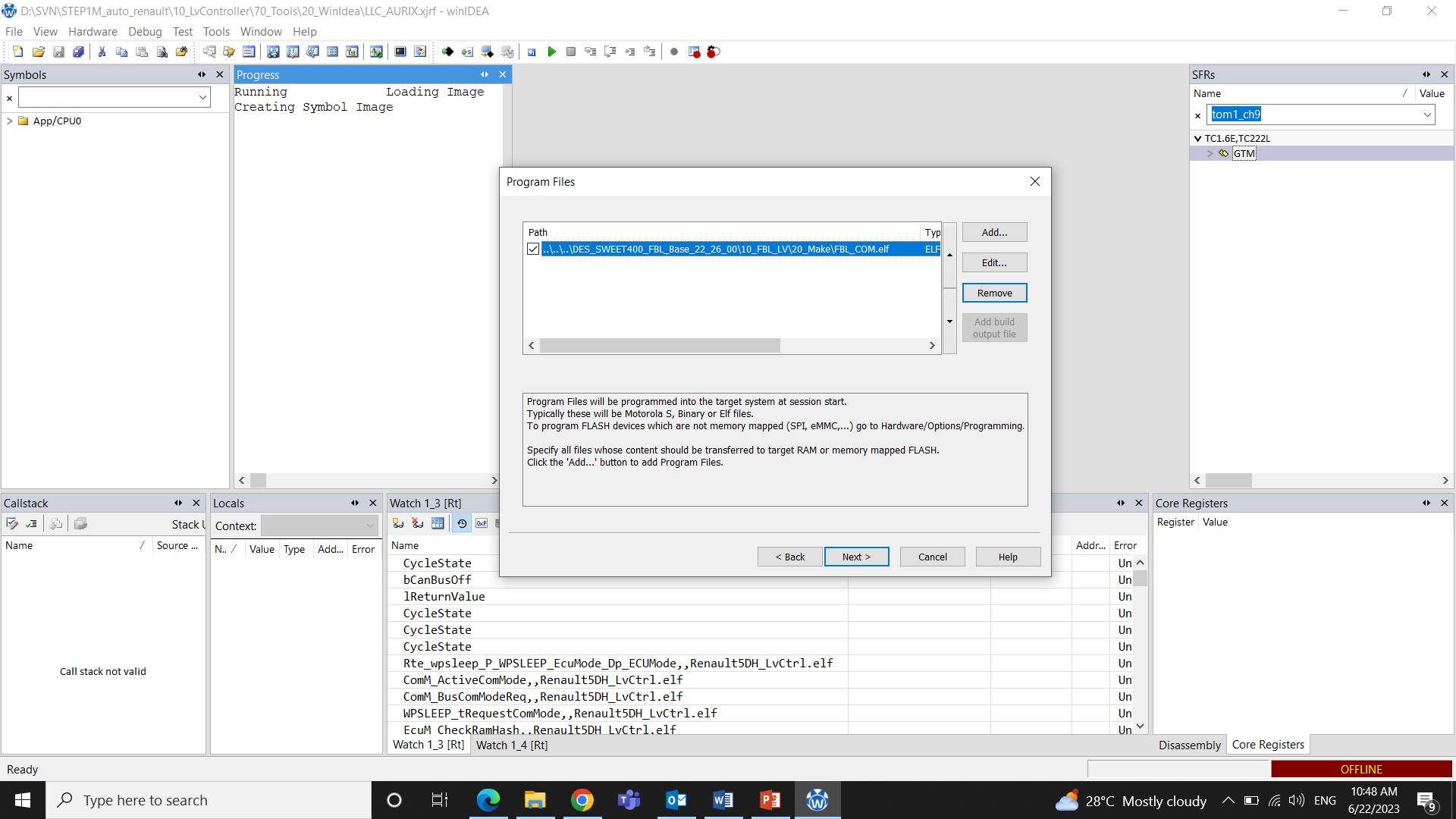
Winidea Debugger is connected to ECU through JTAG and to computer through USB

CAN is connected to computer through USB and to ECU through connector.

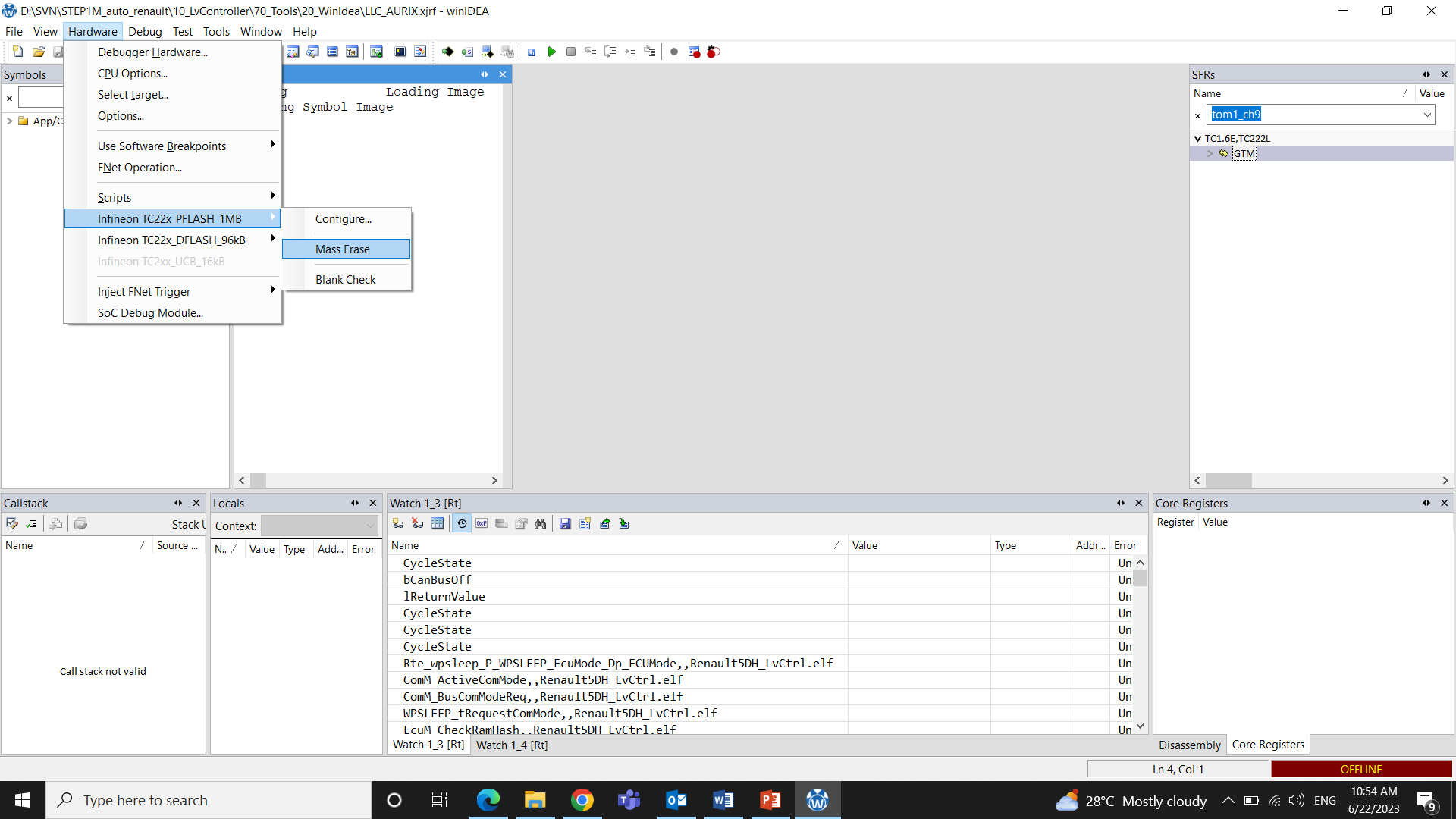
Step 1**:** Repeat the steps as in flashing the Application from Step1 to Step 4

Step 2: In Winidea software select **Debug** > **Config Session** >**Wizard** Add proper. elf file from

D:\SVN\STEP1M\_auto\_renault\DES\_SWEET400\_FBL\_Base\_22\_26\_00\10\_FBL\_LV\20\_Make named **FBL\_COM.elf**

****

Step 3: Mass Erase both P and D flash so that the Soc is without any data



Step 4: In Winidea **Download** and **Play** . Open CANoe and **Start** to see the traces